Assignment1_solutions

October 9, 2024

1 Assignment 1

1.1 GEOS 300

Term 1 (Autumn 2024)

University of British Columbia

1.1.1 Instructor:

Marysa Laguë, Assistant Professor, Department of Geography

1.2 Instructions:

It is strongly recommended that you complete this assignment in Python or R. Templates (this document) are provided for the Python programming language. You may choose to complete this assignment using other software, such as Excel, Sheets, or Numbers.

Please upload your completed assignment as a .pdf file to the course Canvas page as a single, well structured report. Include all figures, tables, graphs, code/calculations, and written answers. We recommend completing the assignment within a Jupyter Notebook document (like this one); you can add "cells" for written answers using the Markdown format for the cell. The completed notebook can then be downloaded as a .pdf file (File -> Save and Export Notebook As -> pdf), which you can upload to canvas.

You can choose to instead write your answers in some other document processor (e.g. Word) and paste your figures, code/calculations etc., however, if you choose to do this, please ensure all your code and calculations are legible, and ensure it is clear what language/program was used to perform the calculations. Label the report document with your name, your student number, the course and year. Upload your report to Canvas by the Assignment deadline on the Canvas page. Do not attach a spreadsheet.

Include your student number on every plot you produce.

Include **correct units on all plots and all answers**, where applicable. Label all axes with the appropriate variables and units.

Points per question are indicated in square brackets. This assignment is worth 10% of the final course grade.

Getting started: enter your name and student number

[2]:
```
Student_Name = 'Marysa Lague'
Student_Number = 123456789
print(f'GEOS 300 Assignment 1 Submission for {Student_Name}: {Student_Number}')
```

```
GEOS 300 Assignment 1 Submission for Marysa Lague: 123456789
```

We need to import python "packages" that contain useful functions for the kind of data analysis covered in this assignment.

[3]:
```
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
```

[ ]:
```

```

1.3 Question 1:

[6] points

Unit conversions! Note that negative exponents mean that unit is in the denominator, e.g. kg / m^3 is the same as kg m^{-3}. Both formats are commonly used and both are acceptable here.

The energy transfer associated with heat is quantified as heat flux (e.g. "sensible heat flux" and "latent heat flux": the flow of energy per unit time per unit area.

(a) [2] What are the units for energy, and what is this in base SI units? (Report full unit name and SI unit symbol)

(b) [1] What are the units for time? (Report full unit name and SI unit symbol)

(c) [1] What are the units for area? (Report full unit name and SI unit symbol)

(d) [2] Combine the base SI units for energy, time, and area to show that the units of heat flux are W/m^2 (show steps / show your work)

(write your answer here!)

Answer:

(a) [2] What are the units for energy, and what is this in base SI units?

[1] for Joule [J]

[1] for J = N m = kg m s^{-2} m = kg m^2 s^{-2}

(b) [1] What are the units for time?

[1] for second [s]

(c) [1] What are the units for area?

[1] for square meter [m^2]

(d) [1] Combine the base SI units for energy, time, and area to show that the units of heat flux are W/m^2:

[1] for multiplying all the base units, [1] for correct answer

(kg m^2 s^{-2})(s^{-1})(m^{-2}) = kg s^{-3}

But if you don't simplify the $m^2 m^{-2}$, you can write:

(kg m^2 s^{-2})(s^{-1})(m^{-2}) = J (s^{-1})(m^{-2}) = W m^{-2}

1.4 Question 2

[2] The density of water is 1000 kg/m^3. Rainfall rate is often measured as a depth of rain, in mm/s, while evaporation is often measured as the mass of water (kg) that evaporates over 1 square m (m^2) each second (s), i.e. $kg/m^2/s$. What is the equivalent of 1 mm/s of rain in the units of $kg/m^2/s$?

Answer:

[1] for showing work, [2] for correct answer (1 mm / s = 1 kg / m^2 / s)

$$1\frac{\text{mm}}{\text{s}} = 1\frac{\text{mm}}{\text{s}} \times \frac{1}{1000}\frac{\text{m}}{\text{mm}} = \frac{1}{1000}\frac{\text{m}}{\text{s}} \tag{1}$$

$$= \frac{1}{1000}\frac{\text{m}}{\text{s}} \times 1000\frac{\text{kg}}{\text{m}^3} = \frac{1000}{1000}\frac{\text{kg m}}{\text{m}^3\text{s}} \tag{2}$$

$$= 1\frac{\text{kg}}{\text{m}^2\text{s}} \tag{3}$$

[ ]:

1.5 Question 3

[3] The atmosphere exerts downwards force on Earth's surface. Following Newton's 2nd law of motion, force is equal to mass times acceleration ($F = ma$).

(a) [1] The SI unit for force is a "newton". From Newton's second law, what are the SI base units of force?

(b) [2] A "pascal [Pa]" is the SI unit for pressure, which is has units of force acting per unit area. Write an expression for pascals using newtons, and a separate expression for pascals using only SI base units.

[ ]:

Answer:

(a) Force [N] = mass [kg] acceleration [m/s^2]
N = kg m / s^2

(b) pressure
[Pa] = force [N] / area [m^2]
Pa = kg m / s^2 / m^2 = kg / m / s^2

---

2 Data Analysis Section

For the rest of the questions, we will be analysing radiation data measured on the UBC Vancouver campus at Totem Field. This data can be found in the file data20100710_py.csv on the Canvas webpage for this assignment. The .csv file contains the following variables: incoming and reflected short-wave radiation (K↓, K↑), incoming and outgoing longwave radiation (L↓, L↑), air temperature (Ta) and relative humidity (RH). Use this dataset to answer the remaining questions. Place the .csv file in the same folder as this .ipynb folder in your JupyterOpen file system.

To help you get started, we have provided a chunk of code below that loads and plots the data. We *strongly encourage* you to step through each line of the code and make sure you understand what is happening (tip: leaving comments in your code is very helpful both for whoever is grading your assignment, but also to remind yourself what you are doing at each step).

First, we need to load the data:

[4]:
```
# Import the data  - upload this file from Canvas and put it in the same folder␣
 ↪as your assignment.
# data_file = 'GEOS300/Fall2024/Assignment1/FromSara/data/data20100710.csv'
data_file = 'data20100710_py.csv'

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

# Pandas (pd here) allows us to set a timestamp as an index which lets us␣
 ↪easily parse time series data
# It opens the csv file into a data format called a "data frame", so we're␣
 ↪going to call it "df" for short
df = pd.read_csv(data_file,parse_dates=['Date'],date_format='%Y-%m-%d %H:
 ↪%M',index_col=['Date'])

# df contains all the variables that were column headers of the .csv file.
# the "Date" dimension is the "index" dimension for the dataframe. The dates␣
 ↪are saved as "datetime" objects which know
# lots of useful things about time, like how to interpret minutes and hours,␣
 ↪etc.

# We can get a extra variables (DOY & HOUR) that will be helpful later
df['HOUR'] = df.index.hour
```

```
df['DOY'] = df.index.dayofyear
df['TIME'] = df.index.time

# Take a quick look at the first few entries - the pandas "head()" command␣
 ↪prints out the top of the dataframe that you just opened:
df.head()

# "NaN" stands for "not a number", and is used in datasets to show where there␣
 ↪is no value for the variable at that time.
```

[4]:

Date	K_in	K_out	L_in	L_out	AirT	RH	HOUR	DOY	TIME
2010-07-10 00:10:00	0.0	0.0	383.0	403.2	NaN	NaN	0	191	00:10:00
2010-07-10 00:20:00	0.0	0.0	370.9	400.8	NaN	NaN	0	191	00:20:00
2010-07-10 00:30:00	0.0	0.0	363.1	399.1	19.3	70.2	0	191	00:30:00
2010-07-10 00:40:00	0.0	0.0	355.6	397.5	NaN	NaN	0	191	00:40:00
2010-07-10 00:50:00	0.0	0.0	357.3	397.4	NaN	NaN	0	191	00:50:00

[ ]:

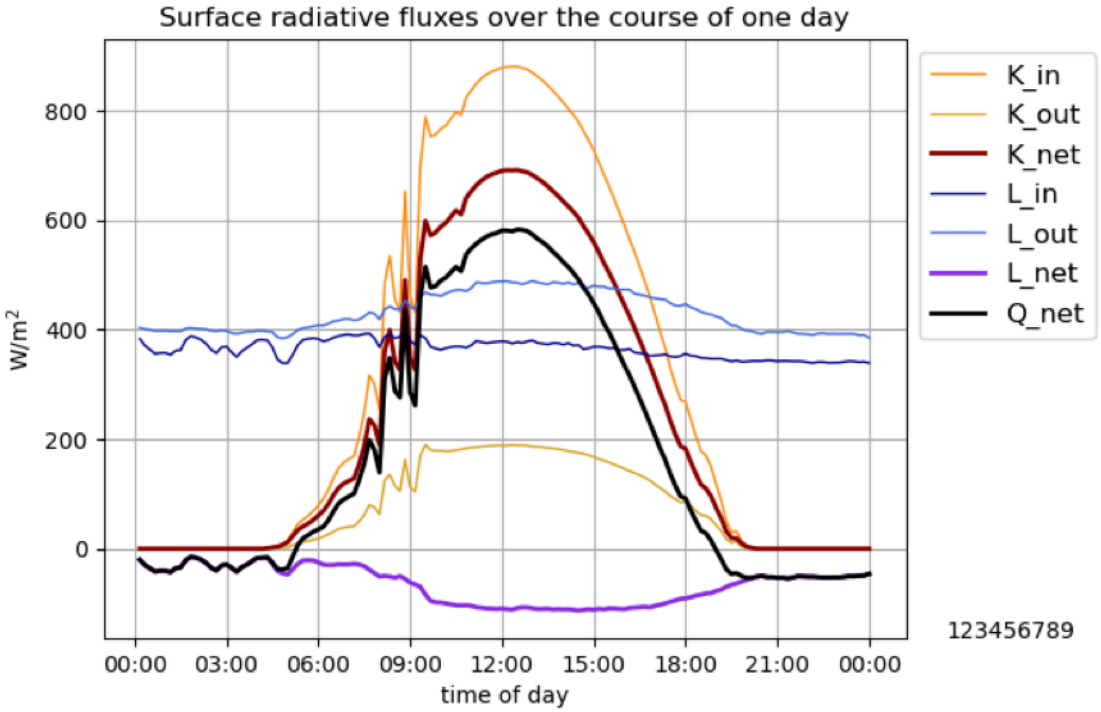
[ ]:

[ ]:

[5]:
```
# Now calculate the net SW absorbed at the surface, the net LW at the surface,␣
 ↪and the net radiation at the surface:
df['K_net'] = df['K_in'] - df['K_out']
df['L_net'] = df['L_in'] - df['L_out']

df['Q_net'] = df['K_net'] + df['L_net']
```

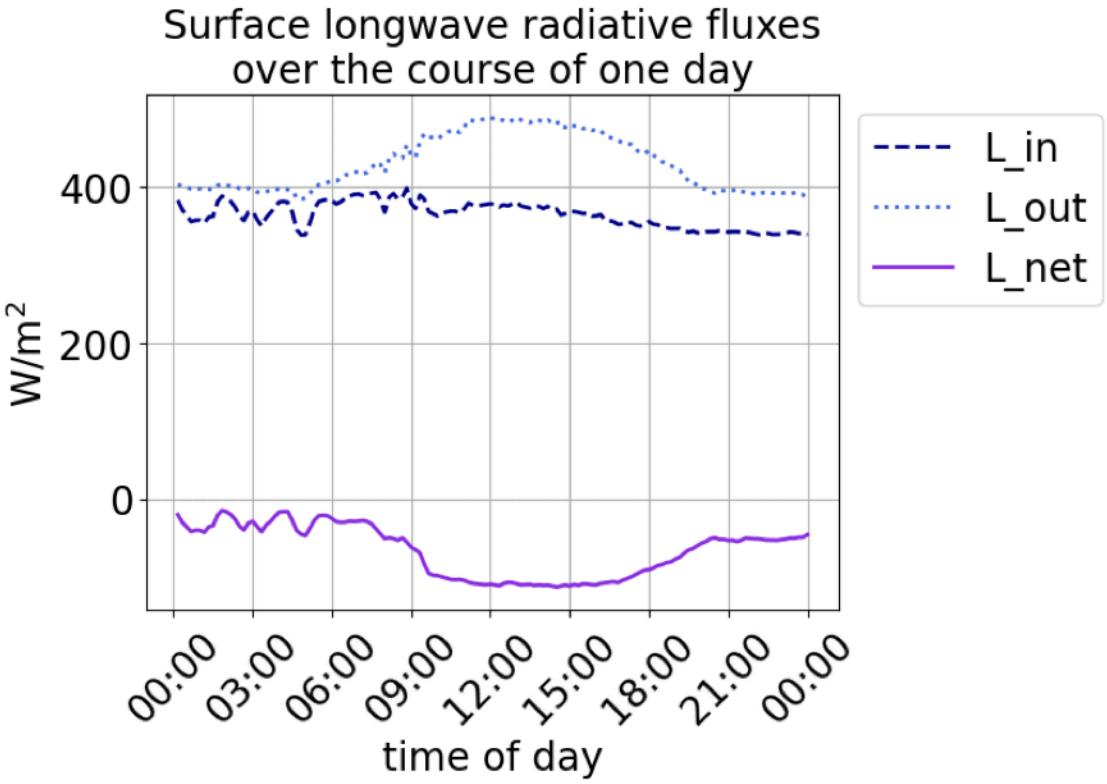
[6]:
```
# Now we'll plot the data to help us answer the question:

plt.plot(df.index,df['K_in'],label='K_in',color='darkorange',linewidth=1)
plt.plot(df.index,df['K_out'],label='K_out',color='goldenrod',linewidth=1)
plt.plot(df.index,df['K_net'],label='K_net',color='darkred',linewidth=2)

plt.plot(df.index,df['L_in'],label='L_in',color='darkblue',linewidth=1)
plt.plot(df.index,df['L_out'],label='L_out',color='royalblue',linewidth=1)
```

```
plt.plot(df.index,df['L_net'],label='L_net',color='blueviolet',linewidth=2)

plt.plot(df.index,df['Q_net'],label='Q_net',color='black',linewidth=2)


# make the plot prettier:
plt.grid()
plt.legend(fontsize=12,loc='upper left', bbox_to_anchor=(1., 1.))
plt.ylabel('W/m$^2$')
plt.xlabel('time of day')
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%H:%M'))

plt.title('Surface radiative fluxes over the course of one day')

# add your student number
plt.text(1.05,0.0,'%1.0f'%Student_Number,
         fontsize=10,transform=plt.gca().transAxes)

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



[11]:
```
# Now we'll plot the data to help us answer the question:

# plt.plot(df.index,df['K_in'],label='K_in',color='darkorange',linewidth=1)
```

```
# plt.plot(df.index,df['K_out'],label='K_out',color='goldenrod',linewidth=1)
# plt.plot(df.index,df['K_net'],label='K_net',color='darkred',linewidth=2)

plt.plot(df.index,df['L_in'],'--',label='L_in',color='darkblue',linewidth=2)
plt.plot(df.index,df['L_out'],':',label='L_out',color='royalblue',linewidth=2)
plt.plot(df.index,df['L_net'],'-',label='L_net',color='blueviolet',linewidth=2)

# plt.plot(df.index,df['Q_net'],label='Q_net',color='black',linewidth=2)


# make the plot prettier:
plt.grid()
plt.legend(fontsize=20,loc='upper left', bbox_to_anchor=(1., 1.))
plt.ylabel('W/m$^2$',fontsize=20)
plt.xlabel('time of day',fontsize=20)
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%H:%M'))
plt.xticks(rotation=45)
plt.title('Surface longwave radiative fluxes\nover the course of one␣
 ↪day',fontsize=20)
plt.gca().tick_params(axis='both', labelsize=20)
# add your student number
# plt.text(1.05,0.0,'%1.0f'%Student_Number,
#          fontsize=10,transform=plt.gca().transAxes)

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



[7]:
```
((df['Q_net'].sum()*(60*10)))*1.e-6
```

[7]: 14.819099999999999

[8]:
```
df['Q_net'].mean()*(60*60*24)/10**6
```

[8]: 14.8191

Try tweaking the above code to change the colours of the lines.

You could look at this plot and try to find when R_{net} reached a maximum, and what its value at that time was - but with a bit of code we can find the exact values, as follows:

[9]:
```
# Using your dataset we'll find the index for the time of the day when we␣
 ↪observe (a) the maximum of R_net
ind_max= df['Q_net'].idxmax()
max_val = df['Q_net'][ind_max]
max_time = df['TIME'][ind_max]

# Using this index, find the time when R_net is maximum  and display output␣
 ↪(using the function 'format' to display only the H:M)
print("The time of the day we observe the maximum of Q_net is "+ max_time.
 ↪strftime("%H:%M")+".")
```

```
# Now lets print the maximum value using a different way of stuffing a number␣
  ↪into a string:
print("The value of Q_net at this time is %1.1f W/m2"%max_val)
```

```
The time of the day we observe the maximum of Q_net is 12:30.
The value of Q_net at this time is 582.6 W/m2
```

2.0.1 Now its your turn:

2.1 Question 4:

[4]

Calculate:

(a) the minimum of Q_net [1]

(b) the maximum of K_net [1]

(c) the maximum of L_net [1]

(d) the minimum of L_net [1]

Include units in answers.

[10]:
```
# type your code here
```

Write your answer here

[ ]:
```

```

Answer:

[11]:
```
# (a) minimum of R_net
ind_min= df['Q_net'].idxmin()
min_val = df['Q_net'][ind_min]
min_time = df['TIME'][ind_min]
print("The minimum value of Q_net is %1.1f W/m2"%min_val)

# (b) maximum of K_net
ind_max= df['K_net'].idxmax()
max_val = df['K_net'][ind_max]
max_time = df['TIME'][ind_max]
print("The minimum value of K_net is %1.1f W/m2"%max_val)


# (c) maximum of L_net
ind_max= df['L_net'].idxmax()
max_val = df['L_net'][ind_max]
max_time = df['TIME'][ind_max]
print("The maximum value of L_net is %1.1f W/m2"%max_val)
```

```
# (d) minimum of L_net
ind_min= df['L_net'].idxmin()
min_val = df['L_net'][ind_min]
min_time = df['TIME'][ind_min]
print("The minimum value of L_net is %1.1f W/m2"%min_val)
```

```
The minimum value of Q_net is -54.5 W/m2
The minimum value of K_net is 690.9 W/m2
The maximum value of L_net is -14.8 W/m2
The minimum value of L_net is -112.9 W/m2
```

[ ]:

2.2 Question 5:

[4]

(a) What is the average value of Q_net over the course of the day? [1]

(b) What surface fluxes balance Q_net at any given point on the land surface? [3]

[12]:
```
# (a) mean of Q_net
mean_val = df['Q_net'].mean()
print("The mean value of Q_net is %1.1f W/m2"%mean_val)
```

```
The mean value of Q_net is 171.5 W/m2
```

Answer:

[ ]:

(b) Q_net is balanced by surface fluxes of sensible heat, latent heat, and ground heat flux. [2] ([1] for all three, [1] for at least 1 of the 3).

[ ]:

2.3 Question 6:

[8]

(a) [4] Calculate the average net short-wave K∗, net long-wave L∗, and net all-wave Q∗ radiative flux densities in W/m2 over the 24 hour cycle.

(b) [4] Then determine the daily energy gain (+) or loss (-) for each flux in (a) by converting the average W/m2 into daily totals (energy per square metre and day, expressed in MJ / day / m2).

Include units in all answers.

[ ]:

[ ]:

Answer:

[3] for values, [1] for units:

[13]:
```
# Averages:

K_mean= df['K_net'].mean()
print("The average value of K_net is %1.1f W/m$^2$"%K_mean)

L_mean= df['L_net'].mean()
print("The average value of L_net is %1.1f W/m$^2$"%L_mean)

Q_mean= df['Q_net'].mean()
print("The average value of Q_net is %1.1f W/m$^2$"%Q_mean)
```

```
The average value of K_net is 237.9 W/m$^2$
The average value of L_net is -66.3 W/m$^2$
The average value of Q_net is 171.5 W/m$^2$
```

[3] for values, [1] for units:

[14]:
```
# Daily totals:

seconds_per_day = 60.*60.*24. # s /day
Joules_per_MJ = 10**6 # J/MJ

K_tot= K_mean*seconds_per_day/Joules_per_MJ
print("The total daily K_net is %1.1f MJ / m$^2$ / day"%K_tot)

L_tot= L_mean*seconds_per_day/Joules_per_MJ
print("The total daily L_net is %1.1f MJ / m$^2$ / day"%L_tot)

Q_tot= Q_mean*seconds_per_day/Joules_per_MJ
print("The total daily Q_net is %1.1f MJ / m$^2$ / day"%Q_tot)
```

```
The total daily K_net is 20.6 MJ / m$^2$ / day
The total daily L_net is -5.7 MJ / m$^2$ / day
The total daily Q_net is 14.8 MJ / m$^2$ / day
```

[ ]:

[ ]:

[ ]:

2.4 Question 7:

[2]

Why do you think the diel cycle (the day-night cycle) of L_in smaller than the diel cycle of L_out?

Write your answer here

Answer:

L_out increases during the day due to solar heating from the sun; as the land warms from K_in, it heats up, leading to increased L_out. L_in from the atmosphere doesn't change dramatically because L_in is a function of the temperature of the whole troposphere; while near-surface air changes temperatures over the diel cycle, the full depth of the troposphere has a smaller diel cycle in temperature, and thus there is a small diel cyclQ4. Calculate solar declination δ for your daye in L_in reaching the surface.

[1] for L_out being influenced by S_in more than L_in

[1] for S_in has a big daily cycle

2.5 Question 8:

[4]

Calculate solar declination δ for the day of the observations.

[15]:
```
### First estimate DOY
DOY = df.DOY

# Estimate gamma ( ) - the fractional year. Note that we are using radians␣
 ↪rather than degrees.
gamma = (2*np.pi/365)*(DOY-1)


# Next, estimate the declination angle using the more precice method from the␣
 ↪lecture slides. We will call this variable delta2
delta = 0.006918 - 0.399912*np.cos(gamma) + 0.070257*np.sin(gamma) - 0.
 ↪006758*np.cos(2*gamma) + 0.000907*np.sin(2*gamma) - 0.002697*np.cos(3*gamma)␣
 ↪+ 0.00148*np.sin(3*gamma)

# Note that delta2 is in radians - we need to convert radians to degrees
delta2deg = delta*(180/np.pi)

print("The declination for my day is: %1.2f"%delta2deg.values[0]+chr(176))
```

```
The declination for my day is: 22.35°
```

2.5.1 simple calculation method:

[16]:
```
### First estimate DOY
DOY = df.DOY

delta_simple = -23.4 * np.cos(2*np.pi * (DOY + 10) / 365 )

print("The declination using the simple calculation for my day is: %1.
 ↪2f"%delta_simple.values[0]+chr(176))
```

```
The declination using the simple calculation for my day is: 22.22°
```

[ ]:

Write your answer here

Answer:

[3] for trying to code it

[1] for correct answer

2.6 Question 9:

Calculate the local apparent time (LAT) for sunrise and sunset.

Hint: Set solar altitude to $= 0\circ$ (for sunrise and sunset) and solve for the hour angle h when $\beta = 0$. Then convert h to an actual time. Note that LAT always ensures solar noon is at 12:00. You will need to use the declination from Q6.

This radiation data was collected on the University of British Columbia campus, located in Vancouver, BC (49.2 ∘ N, 123.2 ∘ W).

Write the equation you use to calculate LAT for sunrise and sunset using declination, solar altitude, and location [4], then calculate LAT using the site data [4].

[8]

[ ]:

Write your answer here

[ ]:

Answer:

Use the equation from lecture YY, slide YY:

$$cos(Z) = sin(\phi)sin(\delta) + cos(\phi)cos(\delta)\cos(h)$$

$$cos(Z) = sin(\beta)$$

We're looking for h.

$$sin(\beta) = sin(\phi)sin(\delta) + cos(\phi)cos(\delta)\cos(h)$$
$$sin(0) = sin(\phi)sin(\delta) + cos(\phi)cos(\delta)\cos(h)$$
$$0 = sin(\phi)sin(\delta) + cos(\phi)cos(\delta)\cos(h)$$
$$-sin(\phi)sin(\delta)/(cos(\phi)cos(\delta)) + = \cos(h)$$
$$h = arccos(-sin(\phi)sin(\delta)/(cos(\phi)cos(\delta)))$$

LAT of sunrise is $ 12 - h/15$, and LAT of sunset is $12 + h/15$.

[17]:
```
phi = 49.2
phi_rad = phi*np.pi/180


delta = 22.35*np.pi/180 # from question 6

cos_h = ( -np.sin(phi_rad)*np.sin(delta) ) / ( np.cos(phi_rad)*np.cos(delta) )

print(cos_h)

h = np.arccos(cos_h)

print(h)

h_deg = h*180/np.pi

# From the slides,  we know h = 15*(12-t), so
# t = 12 + h/15

# LAT is calculated from noon (12); sunrise and sunset occur when beta = 0, in␣
 ↪either direction of local noon.
LAT_sunrise = 12 - h_deg/15
print(LAT_sunrise)

LAT_sunset = 12 + h_deg/15
print(LAT_sunset)


sunrise_time = LAT_sunrise
sunset_time = LAT_sunset


hours = int(sunrise_time)
minutes = (sunrise_time*60) % 60
seconds = (sunrise_time*3600) % 60
```

```
print("The LAT of sunrise: %d:%02d.%02d" % (hours, minutes, seconds))


hours = int(sunset_time)
minutes = (sunset_time*60) % 60
seconds = (sunset_time*3600) % 60

print("The LAT of sunset: %d:%02d.%02d" % (hours, minutes, seconds))
```

```
-0.4763215203262487
2.0672627201288303
4.1036380661255825
19.896361933874417
The LAT of sunrise: 4:06.13
The LAT of sunset: 19:53.46
```

[ ]:

2.7 Question 10:

[2] Vancouver is located in the Pacific Time Zone (UTC -8), which is centered on the 120 W meridian. Using the equations from lecture, calculate the local meant solar time (LMST) and local apparent time (LAT).

[18]:
```
# define LST to be local noon:
LST = datetime.datetime(year=2010,month=7,day=10,hour=12, minute=0, second=0)
print(LST)


# declare the standard meridian of the time zone:
Standard_meridian = -120

# Define the longitude of the station
lon = -123.2

# Define the difference between the station longitude and the standard meridian
delta_lon  = lon-Standard_meridian


# Calculate LMST for both days
LMST = LST + datetime.timedelta(days=0,hours=0,minutes=delta_lon*4,seconds=0)
print(LMST)


print('Local mean solar time (LMST) is ' + LMST.strftime("%H:%M:%S") )
```

```
# Next the time offset between LMST and LAT ( TLAT, i.e. deltaT_LAT), in␣
 ↪minutes can be calculated using the formula given in Lecture 4, Slide 12
# note that we only need gamma for noon, and gamma varies with the day of year␣
 ↪- so fetch a gamma from anything but the last timestep.
# There are fancier ways to get gamma from exactly noon... should be using a␣
 ↪value of gamma=3.27
deltaT_LAT = 229.18*(0.000075 + 0.001868*np.cos(gamma[96]) - 0.032077*np.
 ↪sin(gamma[96]) - 0.014615*np.cos(2*gamma[96]) - 0.040849*np.sin(2*gamma[96]))

deltaT_LAT_temp = 229.18*(0.000075 + 0.001868*np.cos(3.3) - 0.032077*np.sin(3.
 ↪3) - 0.014615*np.cos(2*3.3) - 0.040849*np.sin(2*3.3))
print('temp:')
print(deltaT_LAT_temp)

print(gamma[96])
print('deltaT_LAT:')
print(deltaT_LAT)

deltaT_LAT_datetime = datetime.
 ↪timedelta(days=0,hours=0,minutes=deltaT_LAT,seconds=0)

deltaT_LAT_datetime


LAT = LMST - deltaT_LAT_datetime

print('Local apparent time (LAT) is ' + LAT.strftime("%H:%M:%S") )
```

```
2010-07-10 12:00:00
2010-07-10 11:47:12
Local mean solar time (LMST) is 11:47:12
temp:
-5.345258202844987
3.270699200997593
deltaT_LAT:
-5.089859878992924
Local apparent time (LAT) is 11:52:17
```

```
/tmp/ipykernel_722/3504298967.py:27: FutureWarning: Series.__getitem__ treating
keys as positions is deprecated. In a future version, integer keys will always
be treated as labels (consistent with DataFrame behavior). To access a value by
position, use `ser.iloc[pos]`
  deltaT_LAT = 229.18*(0.000075 + 0.001868*np.cos(gamma[96]) -
0.032077*np.sin(gamma[96]) - 0.014615*np.cos(2*gamma[96]) -
0.040849*np.sin(2*gamma[96]))
/tmp/ipykernel_722/3504298967.py:33: FutureWarning: Series.__getitem__ treating
```

```
keys as positions is deprecated. In a future version, integer keys will always
be treated as labels (consistent with DataFrame behavior). To access a value by
position, use `ser.iloc[pos]`
  print(gamma[96])
```

[ ]:

2.8 Question 11:

[4]

Assume the incoming solar irradiance at the top of the atmopshere ("extraterrestrial erradiance" KEx) above the site at noon on the day of observations is 1178 W/m2 (recall: irradiance from the sun at the solar equator is 1366.5 W/m2, but irradiance at the top of the atmosphere above any given latitude-longitude point on Earth varies with day of the year, latitude, and longitude).

Assume = 63.1 ° (the angle between the surface and the incident sun beam).

What is the approximate bulk transmissivity (Ψa) of the total atmospheric column at this time? Comment upon the reasons for the magnitude of Ψa you find.

[ ]:

Answer:

[4]

[1] for getting most of the code
[1] for getting the right numerical answer
[1] for commenting on how clean/dirty the numerical answer is
[1] for commenting on why there could be dirty air at the time of observations (any plausible explanation is fine here)

A transmisitivity of 0.77 indicates a slightly dirty atmosphere; transmisivity ranges from 0.6 for dirty, smoggy air to 0.9 for clean air. Reasons for a dirty atmosphere over the UBC campus in the middle of July could include polution or forest fire smoke.

[1]:
```
# there are a couple different ways of getting m, which we show here
```

[5]:
```
z = 90-63.1
m = 1/(np.cos(z*np.pi/180))
print(m)
```

```
1.1213307581628822
```

[4]:
```
beta = 63.1
beta_rad = np.pi*beta/180

# Z = 90. - 63.1

# Z_rad = Z*np.pi/180
```

```
sinbeta = np.sin(beta_rad)
print(sinbeta)

m = 1/sinbeta

print(m)
```

```
0.891797529605214
1.1213307581628822
```

[20]:
```
beta = 63.1
beta_rad = np.pi*beta/180

# Z = 90. - 63.1

# Z_rad = Z*np.pi/180

sinbeta = np.sin(beta_rad)
print(sinbeta)

m = 1/sinbeta

print(m)

df_noon = df.loc[df['TIME']==datetime.time(hour=12, minute=0, second=0)]

Kdown_noon = df_noon['K_in']
print(Kdown_noon)

KEx = 1178

print(Kdown_noon)
print(KEx)


psia = (Kdown_noon/KEx)**(1/m)

print('The transmisivity of the atmosphere is approximatley %1.2f'%psia)
```

```
0.891797529605214
1.1213307581628822
Date
2010-07-10 12:00:00    877.7
Name: K_in, dtype: float64
Date
2010-07-10 12:00:00    877.7
Name: K_in, dtype: float64
```

```
1178
The transmisivity of the atmosphere is approximatley 0.77
```

```
/tmp/ipykernel_860/2701710467.py:28: FutureWarning: Calling float on a single
element Series is deprecated and will raise a TypeError in the future. Use
float(ser.iloc[0]) instead
  print('The transmisivity of the atmosphere is approximatley %1.2f'%psia)
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

---

3 END ASSIGNMENT 1 - PUT REST ON ASSIGNMENT 2