Brodzik_assignment1

March 12, 2017

0.0.1 GEOG827

0.0.2 Assignment #1

0.0.3 M. J. Brodzik

0.0.4 due 3/12/17

0.1 Question 1

Using the tables and/or equations in the "Calculating Evaporation" documents posted in blackboard, notes and/or other sources (state the source), express results as mean W/m2 over the day.

0.2 Q1. Part 1

Part i) estimate the daily average solar radiation to the top of the Earth's atmosphere at 56oN on July 2nd.

Total daily solar radiation to top of the atmosphere (TOA) is a function of latitude, season and time of day. From Table D, "Calculating Evaporation Notes", the Total daily solar radiation, K_A $[Wm^{-2}]$, is given in 10-degree latitude increments, for Jun 22 and Jul 15. I begin by plotting these to see how much variability there is in K_A:

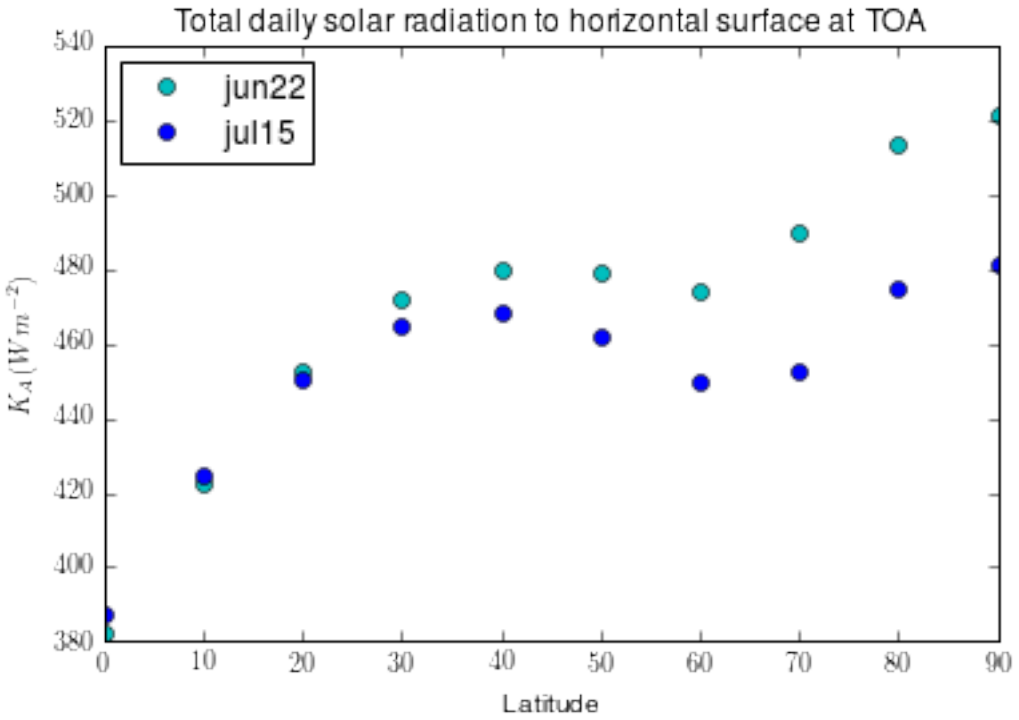
```
In [1]: %pylab inline
        import matplotlib.pyplot as plt
        import numpy as np

        # Define data to plot
        lats = np.arange(10) * 10
        jun22 = np.array([382.68, 422.88, 452.91, 472.29, 480.04,
                          479.07, 474.23, 490.21, 513.46, 521.70])
        jul15 = np.array([387.52, 424.82, 450.49, 465.02, 468.41,
                          462.12, 450.01, 452.43, 474.71, 481.49])

        # Make the plot:
        plt.rc('text', usetex=True)
        fig, ax = plt.subplots(1)
        ax.plot(lats, jun22, 'co', label='jun22')
        ax.plot(lats, jul15, 'bo', label='jul15')
        ax.set_title("Total daily solar radiation to horizontal surface at TOA")
        ax.set_xlabel("Latitude")
        ax.set_ylabel(r'$K_A (W m^{-2})$')
        ax.legend(loc='best', numpoints=1)
        plt.show()
        fig.savefig('HW1.1i_fig1.png')
```

```
Populating the interactive namespace from numpy and matplotlib
```

```
/Users/brodzik/.conda/envs/pmesdr/lib/python2.7/site-packages/matplotlib/font_manager.py:273: UserWarnir
  warnings.warn('Matplotlib is building the font cache using fc-list. This may take a moment.')
```



So although there is an inflection point in the data above 60N, for an estimate I think it's sufficient to just linearly interpolate K_A for the given dates to a value for Jul 2, and then interpolate to 56° N.

```
In [2]: # define a quick linear interpolation function
        # calculate slope and intercept and the value of the line at
        # the new value
        def linear_model_value_at(x, x1, y1, x2, y2):
            slope = (y2 - y1) / (x2 - x1)
            # y = mx + b ==> b = y - mx
            intercept =  y1 - (slope * x1)
            return (slope * x) + intercept
```

```
In [3]: KA_56N_jun22 = linear_model_value_at(56., 50., 479.07, 60., 474.23)
        KA_56N_jul15 = linear_model_value_at(56., 50., 462.12, 60., 450.01)
```

Linearly interpolate K_A at 56 N to July 2 between Jun22 and Jul15:

```
In [4]: import datetime
        jun22_doy = datetime.datetime(2017, 6, 22).timetuple().tm_yday
        jul2_doy = datetime.datetime(2017, 7, 2).timetuple().tm_yday
        jul15_doy = datetime.datetime(2017, 7, 15).timetuple().tm_yday
        print(jun22_doy, jul2_doy, jul15_doy)

        KA_56N_jul2 = linear_model_value_at(jul2_doy,
                                            jun22_doy, KA_56N_jun22,
                                            jul15_doy, KA_56N_jul15)
        KA_56N_jul2
```

```
(173, 183, 196)
```

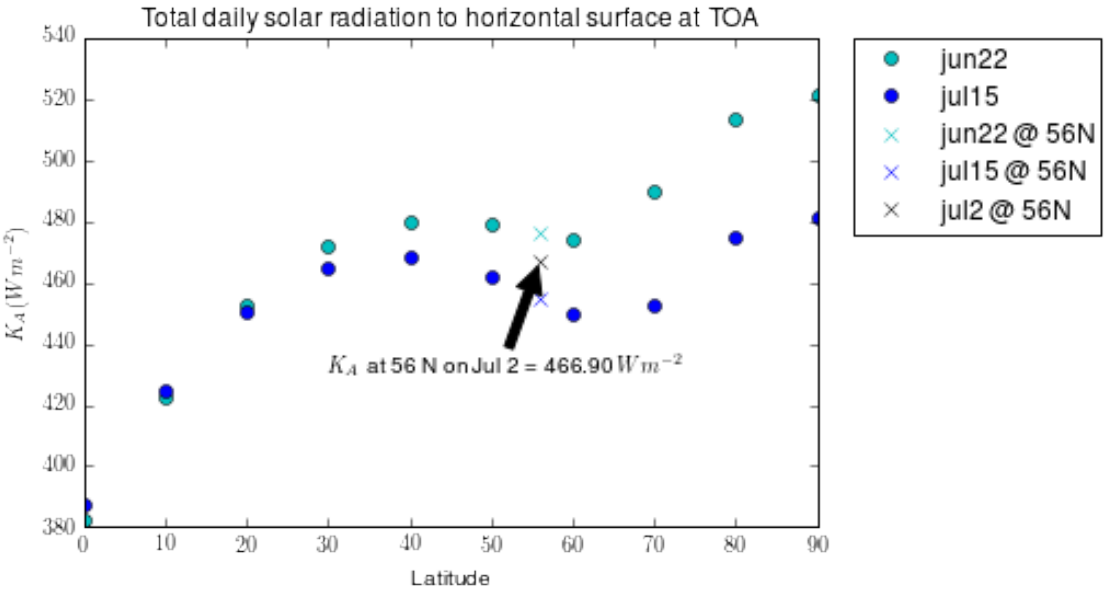
Out[4]: 466.8999130434783

Adding my interpolated values to the plot, I think it's sufficient to use this approximation:

In [5]:
```
fig, ax = plt.subplots(1)

ax.plot(lats, jun22, 'co', label='jun22')
ax.plot(lats, jul15, 'bo', label='jul15')
ax.plot(56, KA_56N_jun22, 'cx', label='jun22 @ 56N')
ax.plot(56, KA_56N_jul15, 'bx', label='jul15 @ 56N')
ax.plot(56, KA_56N_jul2, 'kx', label='jul2 @ 56N')
ax.annotate('$K_A$ at 56 N on Jul 2 = %.2f $W m^{-2}$' % KA_56N_jul2,
            xy=(56, KA_56N_jul2),
            xytext=(30, 430),
            arrowprops=dict(facecolor='k', shrink=0.05))

ax.set_title("Total daily solar radiation to horizontal surface at TOA")
ax.set_xlabel("Latitude")
ax.set_ylabel(r'$K_A (W m^{-2})$')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0., numpoints=1)
plt.show()
fig.savefig('HW1.1i_fig2.png')
```



So the estimated daily average solar radiation to the top of the Earth's atmosphere at 56° N on July 2nd is 466.90 Wm^{-2}.

0.3 Q1. Part ii

ii) if there are 7 hours of bright sunshine, what is the average incoming solar radiation to the surface of a flat unvegetated field at this location on this day?

Eq 7. from "Calculating Evaporation Notes" gives the equation for mean incoming shortwave radiation $K\downarrow [Wm^{-2}]$ as:

$K\downarrow = K_A[a + b(\frac{n}{N})]$

Looking at latitudes of locations in Table B, I think the closest location would be that of Central SK, Canada from Mudiare(1985). I will use the rainfree values for a and b.

Let:

Estimate N from Table C, values given for latitude=56N, interpolate values from Jul1 - Jul5 to a value for Jul2, and remember to convert (hours,minutes) to decimal hours)

```
In [6]: N = linear_model_value_at(2., 1, 17. + (31./60.), 5, 17. + (25./60.))
        print("Estimate of N at 56N on Jul 2 is: %.2f" % N)
```

```
Estimate of N at 56N on Jul 2 is: 17.49
```

```
In [7]: a = 0.27
        b = 0.47
        n = 7.
        K_down = KA_56N_jul2 * (a + b * (n/N))
        K_down
```

```
Out[7]: 213.881978746401
```

So with 7 hours of direct sunshine at this location, the mean incoming solar radiation to the surface of a flat unvegetated field at about 56° N on July 2nd is 213.88 Wm^{-2}. So atmospheric transmittance reduces the shortwave by more than half of that reaching TOA.

0.4 Q1. Part iii

iii) if the mean daily air temperature is 15°C and the mean daily relative humidity is 70%, what is the daily average clear sky incoming longwave radiation and what is the actual daily average incoming longwave radiation to this field?

From our class notes, incoming longwave clear sky radiation from the atmosphere, $L_{0,clear}[Wm^{-2}]$, is expressed by the Stephan-Boltzmann Law:

$L_{0,clear} = \varepsilon_{clear}(T, e)\sigma T^4$

Where apparent clear-sky emissivity, ε_{clear}, is a function of air temperature near the ground, $T[K]$, and water vapour pressure, $e[mb]$, near the ground. Brutsaert(1975) derived an empirical relationship for ε_{clear} as a function of T and e:

$$\varepsilon_{clear} = C(\frac{e}{T})^{1/m}$$

where:

T = air temperature, $[K]$

e = vapour pressure near the ground, $[mb]$

$C = 1.24$

$m = 7$

From lectures, I can use the Magnus Formula to approximate saturated water vapour pressure $e_s[hPa = mb]$, at 15-deg C:

$e_s[mb] = 6.1094 \exp \frac{17.625T}{T+243.04}$

where: T = air temperature $[°C] = 15$.

```
In [8]: T_C = 15. # air temperature, degrees C
        es_15C = 6.1094 * np.exp((17.625 * T_C) / (T_C + 243.04))
        print("Saturated water vapour pressure at 15°C is %.2f mb" % es_15C)
```

```
Saturated water vapour pressure at 15°C is 17.02 mb
```

So water vapour pressure near the ground, $e[mb]$, is 70% of $e_s(T = 15°C)$:

```
In [9]: RH = 70
        e = RH / 100. * es_15C
        print("Water vapour pressure near the ground is %.2f mb" % e)
```

```
Water vapour pressure near the ground is 11.91 mb
```

```
In [10]: # For the Brutsaert equation, temperature needs to be in Kelvins, so convert 15C to Kelvins
         T_K = T_C + 273.15
         print("Air temperature = %.2f K" % T_K)
```

```
Air temperature = 288.15 K
```

```
In [11]: C = 1.24
         m = 7.
         e_clear = C * ((e / T_K)**(1./m))
         print("Clear sky emissivity (using Brutsaert's formula) is %.2f" % e_clear)
```

```
Clear sky emissivity (using Brutsaert's formula) is 0.79
```

So using the Stephan-Boltzmann Law, let:
$\varepsilon_{clear}(T = 15°C, e = 11.91mb) = 0.79$
$\sigma = 5.67$ x $10^{-8} Wm^{-2}K^{-4}$
$T = 288.15\ K$

```
In [12]: sigma = 5.67e-08
         longwave_clearsky = e_clear * sigma * T_K**4
         print("Incoming longwave clearsky %.2f [W/m^2]" % longwave_clearsky)
```

```
Incoming longwave clearsky 307.49 [W/m^2]
```

So I estimate daily average clear sky incoming longwave radiation, $L\downarrow$, to be 307.49 Wm^{-2}, which is greater than the incoming shortwave from part i).

For the actual daily average incoming longwave radiation to this field, since it there are only 7 hours of bright sunshine, I think the actual atmospheric emissivity will need to be adjusted upward, to account for increased emission from the clouds. I will use the adjustment used in Sicart et al 2006, equation (9). The adjustment for emissivity from clouds is:

$$\varepsilon_{cloudy} = \varepsilon_{clear}(1 + 0.44RH - 0.18\tau_{atm})$$

where:
RH = relative humidity (as a percent)
τ_{atm} = shortwave transmissivity of the atmosphere
Shortwave atmospheric transmissivity is the ratio of my answers from part i) and ii):

```
In [13]: tau_atm = K_down / KA_56N_jul2
         print("KA = %.2f, K_down = %.2f, tau_atm = %.2f" %(
                 KA_56N_jul2, K_down, tau_atm))
```

```
KA = 466.90, K_down = 213.88, tau_atm = 0.46
```

So using emissivity of the cloudy atmosphere in Stephan-Boltzmann:

```
In [14]: e_cloudy = e_clear * (1. + 0.44 * (RH/100.) - 0.18* (tau_atm))
         longwave_cloudysky = e_cloudy * sigma * T_K**4
         print("Clear-sky emissivity %.2f" % e_clear)
         print("Cloudy-sky emissivity %.2f" % e_cloudy)
         print("Incoming longwave radiation %.2f" % longwave_cloudysky)
```

```
Clear-sky emissivity 0.79
Cloudy-sky emissivity 0.96
Incoming longwave radiation 376.84
```

So this matches my expectation from class notes that $\varepsilon_{clear} < \varepsilon_{cloudy}$, and I estimate actual daily average incoming longwave radiation to this field as 376.84 Wm^{-2}.

0.5 Q1. Part iv

iv) if the surface temperature is 20°C and albedo is 0.15 what is the average net radiation over the day?

The average net radiation over the day is the sum of net shortwave and net longwave:

$$Q^* = K^* - L^* = K\downarrow(1-\alpha) + L\downarrow - L\uparrow$$

where:
K^* = net shortwave
L^* = net longwave
$K\downarrow$ = incoming shortwave
α = shortwave albedo
$L\downarrow$ = incoming longwave
$L\uparrow$ = outgoing longwave from the terrain surface

Air temperature is not a factor for the incoming shortwave, so I can use incoming shortwave from part ii:
$K\downarrow = 213.88 Wm^{-2}$
$\alpha = 0.15$

```
In [15]: albedo = 0.15
         K_net = K_down * (1. - albedo)
         K_net
```

```
Out[15]: 181.79968193444083
```

And adjust the incoming longwave from part iii for the higher temperature:

```
In [16]: T_C = 20. # air temperature, degrees C
         T_K = T_C + 273.15
         e_clear_20 = C * ((e / T_K)**(1./m)) # clear sky emission at T=20
         e_cloudy_20 = e_clear_20 * (1. + 0.44 * (RH/100.) - 0.18* (tau_atm))
         longwave_cloudysky_20 = e_cloudy_20 * sigma * T_K**4 # W/m^2
         print("Incoming longwave radiation for T=20°C is now %.2f W/m^2" % longwave_cloudysky_20)
```

```
Incoming longwave radiation for T=20°C is now 402.69 W/m^2
```

Outgoing longwave is longwave emission from the terrain surface. I use the Stephan-Boltzmann equation, assuming albedo of the terrain is close to 1, say 0.98, and that the air temperature is an adequate estimate for the surface temperature:

```
In [17]: surface_emissivity = 0.98
         longwave_up = 0.98 * sigma * T_K**4 # W/m^2
         print("Outgoing longwave radiation from surface for T=20°C is %.2f W/m^2" %
               longwave_up)
```

```
Outgoing longwave radiation from surface for T=20°C is 410.36 W/m^2
```

```
In [18]: net = K_net + longwave_cloudysky_20 - longwave_up
         print(K_net, longwave_cloudysky_20, longwave_up)
         print("Net_radiation for T=20 is %.2f" % net)
```

```
(181.79968193444083, 402.69262523294549, 410.3635032136096)
Net_radiation for T=20 is 174.13
```

So I estimate average net radiation in this location over the day to be 159.67 Wm^{-2}.

0.6 Q1. Part v

v) what would the daily average incoming solar radiation and incoming longwave radiation fluxes be to the sub-canopy floor under an adjacent pine canopy at this location on this day, if the pine canopy had a leaf area index of 2.1 and a sky view factor of 0.2? If the sub-canopy surface temperature and albedo are the same as in the field, what is the sub-canopy net radiation?

Per Pomeroy and Dion, 1996, the transmittance through a forest canopy can be modeled with an extinction coefficient:

$$\tau = exp^{\frac{-Q_{ext}\ LAI'}{\sin\theta}}$$

where:
τ = transmittance through forest canopy
Q_{ext} = extinction efficiency (dimensionless)
LAI' = effective winter leaf area index
θ = solar angle above the horizon

From Pomeroy lecture on Day 1, slide 42, apparently Q_{ext} cancels $\sin\theta$, so $Q_{ext} \approx \sin\theta$ so their ratio will $\$ \rightarrow \$1.and\tau$ is:

```
In [19]: eff_lai = 2.1
         forest_tau = exp(-1 * eff_lai)
         forest_tau

Out[19]: 0.12245642825298191
```

So I estimate $\tau = 0.12$. This is consistent with my notes from class, which indicate that a good rule of thumb for τ in forests is 0.1.

For daily average incoming solar radiation and incoming longwave radiation fluxes to the sub-canopy floor under an adjacent pine canopy at this location on this day, incoming shortwave will be K^* times the transmittance, and net shortwave will be incoming shortwave times $(1 - \alpha)$:

```
In [20]: subcanopy_shortwave_down = K_net * forest_tau
         net_subcanopy_shortwave = subcanopy_shortwave_down * (1. - albedo)
         print("shortwave down to subcanopy %.2f W/m^2" % subcanopy_shortwave_down)
         print("net subcanopy shortwave %.2f W/m^2" % net_subcanopy_shortwave)

shortwave down to subcanopy 22.26 W/m^2
net subcanopy shortwave 18.92 W/m^2
```

The incoming longwave to the subcanopy will come from a combination of the sky and the trees. Sicart et al. 2006 (equation 6) found that terrain and vegetation emissions of longwave radiation could be represented as separate components weighted by sky view factor:

$$L = V_f L_0 + (1 - V_f)\varepsilon_s \sigma T_s^4$$

let:
$V_f = 0.2$ (given)
L_0 = longwave from the cloudysky at 20°C from part iv, Wm^{-2}
$\varepsilon_s = 0.98$ emissivity of the forest, assume it is close to 1, from Sicart "terrain emissivity is close to 1 for snow and most natural surfaces"
$T_s = 20 + 273.15\ K$ (same temperature as surrounding field)

The outgoing longwave from the subcanopy floor will be the same as that from the surrounding field.

```
In [21]: sky_view_factor = 0.2
         subcanopy_longwave_down = sky_view_factor * longwave_cloudysky_20 + \
                (1. - sky_view_factor) * surface_emissivity * sigma * T_K**4
         net_subcanopy_longwave = subcanopy_longwave_down - longwave_up
```

```
print("longwave down to subcanopy %.2f W/m^2" % subcanopy_longwave_down)
print("longwave up from subcanopy %.2f W/m^2" % longwave_up)
print("net subcanopy longwave %.2f W/m^2" % net_subcanopy_longwave)
```

```
longwave down to subcanopy 408.83 W/m^2
longwave up from subcanopy 410.36 W/m^2
net subcanopy longwave -1.53 W/m^2
```

So the subcanopy is losing longwave radiation.

```
In [22]: net_subcanopy = net_subcanopy_shortwave + net_subcanopy_longwave
```

Finally, the subcanopy net radiation $Q^* = K^* + L^* = 14.50\ Wm^{-1}$.

0.7 Question 2

Use equations presented in the paper by Harder and Pomeroy (2013) to estimate precipitation phase:

During a precipitation measurement of 5 mm over an hour into a remote unattended Alter-shielded weighing precipitation gauge in a forested clearing you measure an air temperature of +1.0 C, RH of 60%, with very low wind speed. What is the water equivalent depth of snowfall? What is the depth of rainfall?

Harder and Pomeroy (2013) derive a psychrometric equation to derive phase change, expressed as rainfall fraction, f_r, as a sigmoidal function:

$$f_r(T_i) = \frac{1}{1 + b * c^{T_i}}$$

where:
b, c = best fit coefficients, both are functions of time scale of measurement (15-min, hourly, daily)
T_i = hydrometeor temperature

An iterative solution for T_i, Appendix eq. A.5, is:

$$T_i = T_a + \frac{D}{\lambda_t} L(\rho_{T_a} - \rho_{sat(T_i)})$$

where:
T_a = air temperature $[K]$
D = diffusivity of water vapour in air $[m^2 s^{-1}]$
λ_t = thermal conductivity of air $[J\ m^{-1} s^{-1} K^{-1}]$
L = latent heat of vaporisation or sublimation $[J\ kg^{-1}]$
ρ_{T_a} = water vapour density in the free atmosphere $[kg\ m^{-3}]$
$\rho_{sat(T_i)}$ = saturated water vapour density at the hydrometeor surface $[kg\ m^{-3}]$

Since the wind speed is low, I will assume thermodynamic equilibrium.

I am given $T_a = 1.0\ °C = 274.15\ K$. For each of the next terms:

D: Following the appendix equation A.6 (Thorpe and Mason, 1966), I can estimate $D[m^2 s^{-1}]$ as function of $T_a[K]$:

$$D = 2.06 * 10^{-5} * \left(\frac{T_a}{273.15}\right)^{1.75}$$

```
In [23]: C_to_K_offset = 273.15
         Tair_C = 1.0
         Tair_K = Tair_C + C_to_K_offset
         D_m2ps = 2.06e-5 * (Tair_K/273.15)**1.75
         C_to_K_offset, Tair_K, D_m2ps
```

```
Out[23]: (273.15, 274.15, 2.0732159900990022e-05)
```

λ_t: Following appendix equation A.9 (List, 1949), I can estimate λ_t $[Jm^{-1}s^{-1}K^{-1}]$ as function of $T_a[K]$:

$$\lambda_t = 0.000063 * T_a + 0.00673$$

```
In [24]: lambda_t = 0.000063 * Tair_K + 0.00673
         print(D_m2ps, lambda_t, D_m2ps/lambda_t)
```

```
(2.0732159900990022e-05, 0.024001449999999997, 0.0008637878086944757)
```

(I wasn't sure about temperature units in C or K, but my value for the psychrometric exchange ratio, D/λ_t, at 0 °C is consistent with the plot in Figure A1(b)).

L: Since the temperature is > 0 °C, I will use heat of vaporization, $L_v[J\ kg^{-1}]$, equation A.11, as a function of $T_a[C]$:

$$L_v = 1000(2501 - (2.361T))$$

```
In [25]: Lv_Jpkg = 1000. * (2501 - (2.361 * Tair_C))
         print("Latent heat of vaporization = %.2E J/kg" % Lv_Jpkg)
```

```
Latent heat of vaporization = 2.50E+06 J/kg
```

ρ_{T_a}, $\rho_{sat(T_s)}$: To estimate water vapour density $[kg\ m^{-3}]$ in the free atmosphere, I calculate the water vapour pressure, e $[Pa]$, using Dingman (2015), eq 3.9a, p.113:

$$e = \frac{RH}{100} * 611\ exp\left(\frac{17.27\ T}{T + 237.3}\right)$$

where:
RH = relative humidity = 70%
T = air temperature, $°C$

```
In [26]: RH = 60.
         saturated_vapour_pressure_Pa = 611 * np.exp((17.27 * Tair_C)/(Tair_C + 237.3))
         free_air_vapour_pressure_Pa = (RH / 100.) * saturated_vapour_pressure_Pa
         print("saturated vapour pressure [Pa] = %.2f Pa" % (saturated_vapour_pressure_Pa))
         print("free air vapour pressure [Pa] for %d%% RH = %.2f Pa" % (
                 RH, free_air_vapour_pressure_Pa))
```

```
saturated vapour pressure [Pa] = 656.92 Pa
free air vapour pressure [Pa] for 60% RH = 394.15 Pa
```

And then convert pressure to density using the ideal gas law (also from Appendix A.8):

$$\rho = \frac{m_w e}{RT}$$

where:
m_w = molecular weight of water = 0.01801528 $kg\ mol^{-1}$
e = water vapour pressure, kPa
R = Universal Gas Constant 8.31441 $J\ mol^{-1}\ K^{-1}$
T = temperature K

```
In [27]: mw = 0.01801528  # kg mol^-1
         R = 8.31441      # J mol^-1 K^-1
         Pa_per_kPa = 1000.  # conversion
         saturated_vapour_density_kgpm3 = (
             mw * saturated_vapour_pressure_Pa / Pa_per_kPa) / (R * Tair_K)
         free_air_vapour_density_kgpm3 = (
```

```
            mw * free_air_vapour_pressure_Pa / Pa_per_kPa) / (R * Tair_K)
        print("saturated water vapour density = %.2E kg m^-3" % (saturated_vapour_density_kgpm3))
        print("free air water vapour density for %d %% RH = %.2E kg m^-3" % (
                RH, free_air_vapour_density_kgpm3))
```

```
saturated water vapour density = 5.19E-06 kg m^-3
free air water vapour density for 60 % RH = 3.12E-06 kg m^-3
```

Solving for hydrometeor temperature:

```
In [28]: T_i = Tair_C + (D_m2ps/lambda_t) * Lv_Jpkg * (
             free_air_vapour_density_kgpm3 - saturated_vapour_density_kgpm3)
         print("hydrometeor temperature = %.2E deg-C" % T_i)
```

```
hydrometeor temperature = 9.96E-01 deg-C
```

And finally, using b and c from middle plot (hourly) of Fig. 6:

```
In [29]: b = 2.50286
         c = 0.125006
         rainfall_fraction = 1. / (1. + (b * c**T_i))
         precip_mm = 5
         print(rainfall_fraction, rainfall_fraction * precip_mm, \
               ((1 - rainfall_fraction) * precip_mm))
```

```
(0.75999258631298428, 3.7999629315649215, 1.2000370684350785)
```

So given a gauge precip measurement of 5 mm, I estimate it fell as 3.8 mm of rainfall, and 1.2 mm water equivalent from snowfall.

0.8 Question 3

You are measuring air temperature and water vapour content over a natural grassland (roughness length = 0.03 m) to get an idea of how available energy is being used at this site. You have installed the minimum of two levels of measurements of wind speed, temperature and humidity. The daily average measurements are (heights above the ground): - 1-m height, wind speed is 0.6 $m\ s^{-1}$, air temperature 20 ^oC, and vapour pressure 2.0 kPa;
- 2-m height, wind speed is 0.62 $m\ s^{-1}$, air temperature 19 ^oC, and vapour pressure 1.5 kPa.

Assume Pa = 101.3 kPa, ρ_a = 1.2 $kg\ m^{-3}$, and c_p = 1005 $J\ kg^{-1}K^{-1}$ and L_v = 2.454 $MJ\ kg^{-1}$.
(a) Estimate a z_0 for momentum, heat and water vapour from the vegetation height. Then using bulk transfer flux-gradient calculations (example in Helgason and Pomeroy, 1995) and ignoring stability corrections find Q_E and Q_H.
(b) Calculate the footprint representative of these measurements which accounts for 80% of the flux. Recall that you will first need to calculate u*.

Per Helgason and Pomeroy, 2005, a typical method to estimate momentum roughness length z_{0m} is to plot $ln(z)$ vs. $\bar{u}$ for neutral conditions and then determine the value of z_{0m} as the y-intercept where $\bar{u}$ =0.

```
In [62]: import numpy as np

         u1 = 0.6  # m/s
         u2 = 0.62 # m/s
         z1 = 1.0  # m
         z2 = 2.0  # m

         # quick linear interpolation function
```

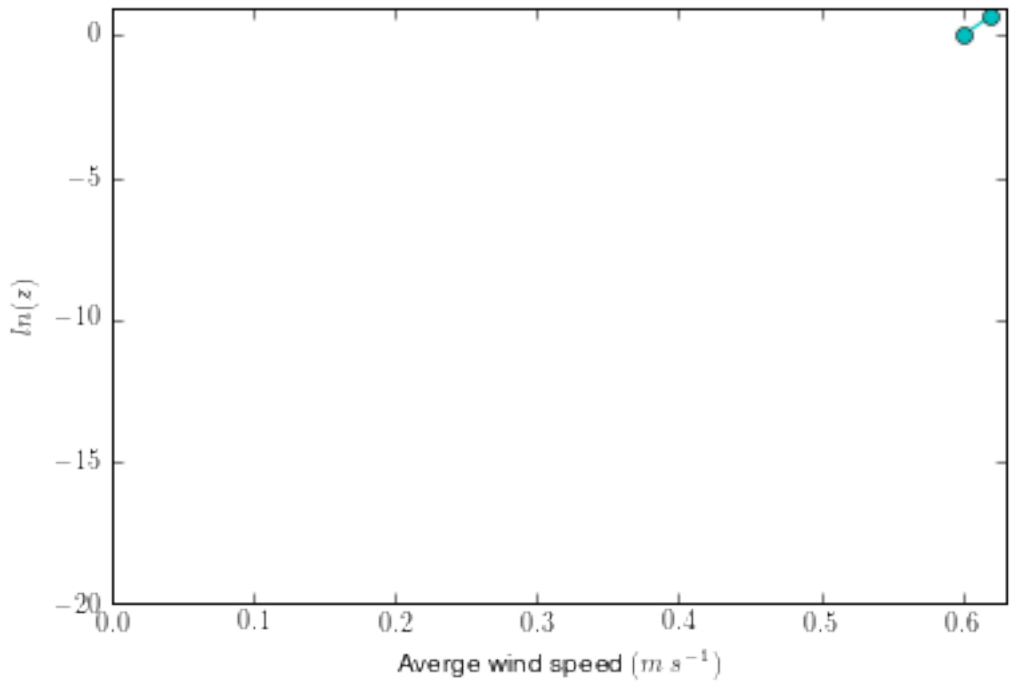
```
        # returns slope and intercept
        def linear_model_parms(x1, y1, x2, y2):
            slope = (y2 - y1) / (x2 - x1)
            # y = mx + b ==> b = y - mx
            intercept =  y1 - (slope * x1)
            return {'m':slope, 'b':intercept}

        parms = linear_model_parms(u1, np.log(z1), u2, np.log(z2))
        print("y-intercept: ln(z) = %f" % parms['b'])

        # Make the plot:
        plt.rc('text', usetex=True)
        fig, ax = plt.subplots(1)
        ax.plot([u1, u2], [np.log(z1), np.log(z2)], "co-")
        #ax.set_title("£ln(z) vs. \bar{u}")
        ax.set_xlabel('Averge wind speed $(m\ s^{-1})$')
        ax.set_ylabel("$ln(z)$")
        ax.set_xlim([0,0.63])
        ax.set_ylim([-20, 1.0])

        plt.show()
        fig.savefig('HW1.3_fig1.png')
```

```
y-intercept: ln(z) = -20.794415
```



Solving for the corresponding height:

```
In [71]: print("Momentum roughness length z_0m = %e meters" % np.exp(parms['b']))
```

```
Momentum roughness length z_0m = 9.313226e-10 meters
```

So my calculated momentum roughness length is about 10 Angstroms, which seems impossibly small. But I am unsure as to what I've done wrong, here.

Taking a different approach, the problem states that the roughness length is 0.03 m. Using Dingman eq (3.28 and 3.29), p. 122, if this roughness length is z_0, then the average vegetation height z_{veg} = 0.3 m = 30 cm , which seems reasonable for a "natural grassland". And, the zero-plane displacement height, z_d is therefore 0.7 * 0.3 = 0.21 m.

Using equation 3.30b to estimate friction velocity, u^*:

$$u^* = \frac{\kappa[u(z_2) - u(z_1)]}{ln(\frac{z_2 - z_d}{z_1 - z_d})}$$

where:
$\kappa = 0.4$

```
In [74]: z_veg = 10. * 0.03
         z_d = 0.7 * z_veg
         kappa = 0.4
         friction_u = (kappa * (u2 - u1)) / np.log((z2 - z_d) / (z1 - z_d))
         print("friction velocity u* = %f m/s" % friction_u)
```

```
friction velocity u* = 0.009781 m/s
```

So I estimate the friction velocity $u^* = 0.0098\ m\ s^{-1}$

Using Dingman eq. 3.35, the momentum flux is:

$$F_M = -\rho_a\ u^{*2}$$

Letting
$\rho_a = 1.2\ kg\ m^{-3}$
u^* = friction velocity $m\ s^{-1}$

```
In [77]: # Units kg/m3 * m2/s2 = kg/m/s2  ?? flux units make sense,
         #                                  kg m/s per unit area per unit time
         rho_a = 1.2 # kg/m-3
         momentum_flux = -1 * rho_a * friction_u
         print("Momentum flux is %f kg/m/s^2" % momentum_flux)
```

```
Momentum flux is -0.011737 kg/m/s^2
```

To calculate latent heat flux Q_E, I think I would need to use Dingman eq 3.47, but I've gotten thoroughly confused and don't understand what z_m is supposed to represent.

To calculate sensible heat flux Q_H, I think I would need to use Dingman eq 3.57, but again, I can't figure out what z_m represents.

(Problems 4 and 5 not attempted.)

```
In [ ]:
```