BEE313 Problem 2.1

Objective

The goal of this problem is to determine how long it takes for the snow pack on Hogg Pass' SNOTEL station to fully melt. After it is known that there is no more snow after April 1st.

Equation:

Total Energy to Fully Melt a SnowPack Utotal = Ucc + Ur + Uout

Ucc = ci *rho_w* h_swe *(T_mp - T_s) # energy to get to 0C which is melting point

Ur = hs *theta_wmax* rho_w * lambda_frz

U_output = (h_swe - hs*theta_wmax)*rho_w*lambda_frz

Time to warm snow pack = Ucc/FE, FE = net energy flux

Time to Ripen = Ur/FE

Tiome to Complete Output Stage = Uoutput/FE

Methods:

The known parameters given by the problem statement were documented. After which point the energy for warming was calculated, and thus the time to warm was then found. The next step was to determine the ripening time. The max water that could be held within the snow pack was determined, and then used to find the ripening energy. This ripening energy was then in turn utilized to calculate the time to ripen. Finally the output energy from the water released from the snow pack after the maximum water held within the snow pack had been reached was computed to find it's associated time. The three times were summed in order to compute a total time.

In [ ]:
```
# Import statements
import numpy as np
import math
from datetime import timedelta, datetime
# known parameters:
## Net Energy Flux
FE = 5.3 # MJ/((m^2)*(day))
## specific heat of ice
ci = 2120/1000000 #MJ/kgC
## snowpack depth
hs = 62 * 0.0254 # converted to meters

## density of water
```

```
rho_w = 1000 #kg/m^3, will change based on temperature...
## Temp Melting Point
Tmp = 0 # Celcius
## Temp Snow Pack
Ts = -4.5 # Celcius
## snow water equivalent
h_swe = 16 * .0254 # meters

# Liquid Water Content Max
### calc max where we use the snowpack density
rho_s = (rho_w * h_swe)/ hs#kg/m^3 at 0C
theta_wmax = (3*10**-10)*(rho_s**3.23) ## Density of ice or snow here????
# latent heat of freezing
lambda_frz = .344 #MJ/kg

print("The Times at Each Stage")
print(" ")
# Calculated time to Warm
Ucc = ci * rho_w * h_swe *(Tmp - Ts)
time_warm = Ucc/FE # days
print(f'Time to warm {time_warm:.3f} days') # correct....

# Calculated Time to Ripen
## total energy required to complete ripening
Ur = hs * theta_wmax * rho_w * lambda_frz
## time to complete ripening
time_ripe = Ur/FE
print(f'Time to ripen {time_ripe:.3f} days')

# Calculated Time to Output:
U_output = (h_swe - hs*theta_wmax)*rho_w*lambda_frz
time_output = U_output/FE
print(f'Time to output {time_output:.3f} days')

print("\n")


# Rounded to the Next Whole Day
days_melt = math.ceil(time_warm+ time_ripe + time_output)
print(f"The time to melt the snow pack was foudn to be {math.ceil(days_melt)} days")

# Added the 28 days to the inital date using datetime
Start_Date = "4/01"

Melt_Date = (datetime.strptime(Start_Date, "%m/%d") + timedelta(days =days_melt)).str
print(f"Date that snowpack will be fully melted by: {Melt_Date}")
```

```
The Times at Each Stage

Time to warm 0.732 days
Time to ripen 1.890 days
Time to output 24.487 days


The time to melt the snow pack was foudn to be 28 days
Date that snowpack will be fully melted by: 04/29
```

Results:

It was determined that on April, 29th, that the snowpack at Hogg Pass' Snotel would fully melt.

Discussion:

The snowpack itself was roughly 1.56 meters in depth. This is an extremely thick layer, however with regard to density it was around 258 kg/m^3 indicating a fresh snow layer. Even though this was a relatively new snowpack, the time to melt it would take a little over 27 days. Indicating that it would not be until the 28th day on April, 29th that the snow pack would have melted. It is important to note that based on the relationships between the density of the snow pack and the warming, ripening, and output stages. An increase in the density of the snow pack would not necessarily indicate an increase in melting time. In fact this would have no change. This is due to the relationships the other associated variables have that are directly related the the density and increase as such.

BEE313 Problem 2.2

Objective :

The goal of this problem was to determine the fraction of net incoming radition that was converted to latent energy. This was based on the fact that there was no ground heat flux, water advected energy or energy storage at the surface. The field was known to be well watered indicating that the saturated vapor pressure at the surface temperature was equivalent to the vapor pressure at an evaporating surface. The air and field temperatures were recorded to be at 7.5 and 1.2C respectively. While the relative humidity measured at the weather station was at 90%.

Evaporative Fraction = 1/(1+B)

B = ((cp - p)/(.622*lambda_v))*((T_s - T_zm)/(e_s - e_zm))

Methods:

First determined the psychometric constant. Then after this the surface temperature saturated vapor pressure was found and then assumed assumed that this vapor pressure would be the evaporative surface vapor pressure. The saturated vapor pressure was then found at the height of the weather station. The vapor pressure measured at the weather station was then found based on the relationship that the saturated vapor pressure would be based on the current relative humidity at that point. The Bowen ratio was then calculated. After the Bowen ratio was determined the evaporative fraction was calculated based on the predetermined Bowen ratio. The evaporative fraction would then be representative of the latent energy.

```
In [ ]: # define the functions:
        def diff_t(T1, T2):
```

```
    t_dif = T1 - T2
    return t_dif
def diff_e(e1, e2):
    e_dif = e1 - e2
    return e_dif
def psycho(Tevap):
    psy = ((cp*p )/(.622*(2.50-((2.36*10**-3)*Tevap))))
    return psy
# psycho = ( (cp - p )/(.622*lambda_v))
def Bowen(T1, T2, e1, e2):
    B = psycho(T1)*(diff_t(T1, T2)/diff_e(e1, e2))
    return B
# def Bowen():
#     B = psycho()*(diff_t(T_s, T_zm)/diff_e(e_s, e_zm))
#     return
# Variables:
## temperatures
T_s, T_zm = 1.2, 7.5
## psychometric values
cp, p= 1*10**-3, 101.3 #MJ/kg, kPa, MJ/kg

## relative humidity
RH = .9

# calculate value for the surface temperature vapor pressure
e_star_s = .611*np.exp((17.3*T_s)/(T_s+237.3)) # .66
## e_star_s is equivalent to es (evaporative surface vapor pressure)
e_s =  e_star_s
# calculate the value for the vapor pressure a 1m
e_star_1m = .611*np.exp((17.3*T_zm)/(T_zm+237.3))

## Requires relative humidity..
e_zm = RH * e_star_1m

# Use the function for the Bowen Ratio in order to determine the Fraction of Latent Er
B_ratio = Bowen(T_s, T_zm,e_s, e_zm)
Evap_Frac = 1/(1+B_ratio)
print(f"The Fraction of incoming radiation converted to latent energy is {Evap_Frac:.3
```

```
The Fraction of incoming radiation converted to latent energy is 0.395
```

Results

The fraction of incoming radiation was converted to the latent energy is .395.

Discussion

The latent energy was found to be 39.5% of the incoming radiation. However, this energy is not being stored at the surface based on the predetermined assumptions so will likely be reflected off. Also it is important to note that the surface saturated vapor pressure was assumed to be equal to that of the evaporative surface vapor pressure. So, it could be that this value might be slightly different if there were other parameters incorporated into this problem. However, this solution of 39.5% LE is a surface level understanding for the study area with the given site conditions.

BEE313 Problem 2.3

Objective

The goal of this problem is to determine the expected daily evapotranspiration (mm/day) from Crater Lake. The data was collected on a late summer day via a floating weather station. The average air temperature at 2 meters was found to be 21.3 C, and the temperature at the surface of the water was found to be 12.4 C. Additionally the wind speed and relative humidity were found to be 2.6 m/s and 45%. The air pressure is 85% of standard air pressure at the lake. It is also important to note that the the water vapor transfer coefficient cannot be used based on the size of the lake and the air pressure.

KE = (1.2*10**-6)/(ln((zm-zd)/(z_0)))**2

E = K_E*u_m*(e_s-e_zm)

Methods

First solved for the vapor pressure of an evaporating surface (es) by recognizing that it was equal to the saturation vapor pressure at the surface temperature. This was done using the claussius clapeyron relationship. Then the vapor pressure at 2 meters above the surface was computed using the claussius clapeyron relationship with the temperature 2m above the surface. Then the relative humidty was multiplied by this value to determine the vapor pressure at this height. The water vapor transfer coefficient was then solved for using equation 6.11 in Physical Hydrology. Then this value was plugged into the standard equation in order to calculate the evaporation.Finally the wind speed, the water vapor transfer coefficient, and the wind speed were plugged into the equation for Evaporation based on Fick's Law to determine the daily evapotranspiration.

In [ ]:
```
from numpy import log as ln
# ET Question

# Calculate Evaporation (Open Water Assumption...using---- mass transfer method....

# Values:
# dimensionless constant
k = .4
## roughness height
#### See typical value for water
#### Table 6.4
z_0 = .0012
## Measurement Height
zm = 2 # m
## zero-plane displacement height
#### See typically value for water
#### Table 6.4
zd = 0
```

```
## wind speed
u_m = 2.6 # m/s

## relative humidity
RH = .45

## Temperatures
T_2m = 21.3 #C
T_0m = 12.4 #C

# Air pressure
p_a = 101.3 #kPa
p_lake= p_a*.85
rho_a = 1.220 #kg/m^3
rho_w = 1000 #kg/m^3

# Step 1 is to determine the 2m vapor pressure using classious clapeyron
e_star_2m = .611*np.exp((17.3*T_2m)/(T_2m+237.3)) # kPa
# Step 2 then plug this value into the equation for vapor pressure at height z
## Requires relative humidity..
e_zm = RH * e_star_2m
# Step 3 then we know that X is our vapor pressure at e(z) and that the air pressure i
### so the air pressure is 85% of what <- assume vapor pressure to determine surface w
### Saturated Vapor Pressure at Surface
e_star_0m = .611*np.exp((17.3*T_0m)/(T_0m+237.3))
### Saturated Vapor Pressure at Surface is Equal to the Vapor PRessure of an Evaporati
e_s = e_star_0m
# Use EQuation 6.11 to solve for K_E
K_E = (.622*(k**2)*rho_a)/(rho_w*p_lake*(ln((zm-zd)/(z_0)))**2)

#  Use Equation 6.12 to solve for E
E = K_E*u_m*(e_s-e_zm) # meters/second

# converted into mm/day
E_mm_day = E*3600*24*1000 #mm/day
print(f"Thedaily evapotranspiration from Crater Lake {E_mm_day:.3f} mm/day") # Work Se
```

```
Thedaily evapotranspiration from Crater Lake 1.724 mm/day
```

Results

It was determined that the daily evapotranspiration from Crater lake was 1.724 mm/day.

Discussions

If the air pressure was not 85% of standard air pressure due to elevation then the simplified formula to determine the water vapor transfer coefficient could have been utilized, and the area of the lake may have been utilized in this calculation. The However, based on the current scenario the area of the lake can be disregarded. So if it was a lower lying lake it is likely that this assumption could have been made. The value of 1.724 mm/day seems like a reasonable value for assumed daily evaporation from a lake. In a prior study conducted in 2002 in Mountatin Lakes, it was determined that average daily evaporation from Lake Laka ranhed from 1.8 to 3.40 mm/day (Pérez et al., 2020). It is likely that there is a source of error in the assumptions made in the calculations for this problem. The fact that values of a lake at a high elevation had similar evaporations though in a different climate in the Mediterranean is reassuring.

BEE313 Problem 2.4

A new variety of potatoes has recently been developed that uses less water than traditional types. This new crop is planted in April and harvested in August. It has the following monthly crop coefficient values (April through August): Kc = [0.2, 0.5, 0.9, 0.6, 0.3]. Data from the local Agrimet station (HERO) gives the estimated monthly reference evapotranspiration for these months as: ETr = [124, 185, 233, 284, 245] in mm/month. You are asked to estimate the amount of irrigation water required to grow this new potato variety in Hermiston, OR. Assume that precipitation is minor and that irrigation during each month must match crop water use.

In [ ]:
```
# ET for specific Crop type... potato
# ETr is the reference crop evap transpriation...

# Etr * crop coefficient

Kc_array = np.array([0.2, 0.5, 0.9, 0.6, 0.3])
ETr_array = np.array([124, 185, 233, 284, 245])

ETc_array = Kc_array*ETr_array
print(ETc_array)
water = np.sum(ETc_array)
print(f"The estimated amount of irrigation from April to August is {water} mm")
# Estimate amount of irrigation water required to grow this new potato variety,....
```

```
[ 24.8  92.5 209.7 170.4  73.5]
The estimated amount of irrigation from April to August is 570.9 mm
```

Results:

The estimated amount of irrigation required from April to August is 570.9 mm

Discussion:

It was determined that in total 570.9mm of water would need to be applied to the crops. However, it is important to note that this can not be evenly distributed over the 5 months, and that based on the resulting array, the schedule needs to follow the product of the reference ET and crop coefficient. This is based on the seasonal differences in temperature that impact the rates of evapotranspiration. With this method there can be an efficient irrigation system put in place that would allow for improved yields and reduced water loss.

References:

Pérez, A., Lagos, O., Lillo-Saavedra, M., Souto, C., Paredes, J., & Arumí, J. (2020). Mountain Lake Evaporation: A Comparative Study between Hourly Estimations Models and In Situ Measurements. Water, 12(9), 2648. https://doi.org/10.3390/w12092648