Howard University PBL Workshop Computational Warmup Exercises

This set of exercises is by no means meant to be comprehensive of the techniques that you might need to accomplish the work during the workshop but if you are comfortable with these exercises (please do them to find out!) then you have a good starting level of computational skills. This document is prepared in Mathematica so you can work on doing the same things in your own computational language like Python, Matlab or R. If, by chance, you are using Mathematica you cannot just copy the statements here. I expect you to be able to explain how each of these statements operates and to come up with your own versions.

Exercise 1 (HDF file import and data manipulation).

The datafile in question is processed from a Multi - Filter Rotating Shadowband Radiometer (MFRSR) which measures radiant energy in 7 spectral bands only 5 of which are used here) permitting calculations of various quantities including aerosol and cloud optical depths. For this datafile, the tasks in order are :

1) assess the various data structures in the HDF file
2) import the aerosol/cloud optical thicknesses measured at 414, 500, 614, 673 and 869 nm
3) use the cloud mask to select just the aerosol optical depths
4) plot up the AODs with a legend to distinguish the plots
5) compute the Ångström coefficient between the 414 and 869 nm wavelengths

First assess the data structure. Since this is an hdf5 file format it is "self-describing". In other words you can interrogate the file to figure out what's in it and you don't have to worry about formatting issues as in an ASCII file such as in Exercise 2. The statement below tells what datasets are contained within the

file. To retrieve this file, go to https://dnwsite.weebly.com/current-measurements.html and look for the Google Drive link in the MFRSR section called “Retrievals”. Drill down through that and find the folder called Retrievals677 where you will find the file.

```
In[34]:= SetDirectory[
           "C:\\Users\\Dave\\Google Drive\\MFRSRProcessed\\SN677\\BVNorthBldg\\Retrievals677\\"];
Out[34]= C:\Users\Dave\Google Drive\MFRSRProcessed\SN677\BVNorthBldg\Retrievals677

In[35]:= Import[MFRFile = "Ret_677_20200227v0.9.h5"]
Out[35]= {/CloudFlag, /SfcPress, /ElevAngles, /IPW, /O3OD, /DerivedO3OD,
          /MFRVolts, /MFRTime, /OptAirMass, /ROD, /MeanLogV0TOA, /AOD}
```

To perform the exercise, we will need the Time, Voltage, CloudFlag and AOD fields which are very simply read due to the hdf file structure.

```
In[16]:= {MFRTime, MFRVolts, OD, CloudFlag} =
           Import[MFRFile, {"Datasets", {"MFRTime", "MFRVolts", "AOD", "CloudFlag"}}];
```

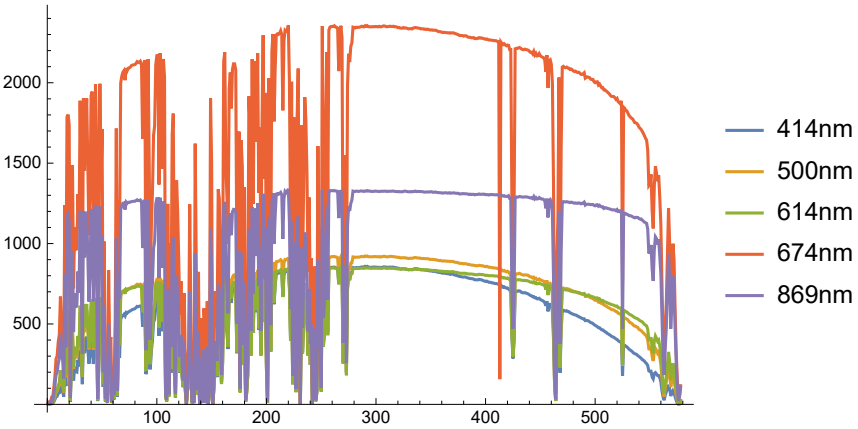
A look at the structure of these datasets indicates that there are 578 temporal samples in the dataset. There are 5 channels of raw voltage data, 5 channels of derived optical depths which includes both clear and cloudy measurements, and a CloudFlag for distinguishing the clear and cloudy cases.

```
In[17]:= Dimensions /@ {MFRTime, MFRVolts, OD, CloudFlag}
Out[17]= {{578}, {5, 578}, {5, 578}, {578}}
```

First look at the raw measurements. The clear parts of the day yield smooth curves while the cloudy parts show large variation.

```
In[18]:= ListPlot[MFRVolts, PlotLegends → {"414nm", "500nm", "614nm", "674nm", "869nm"},
          PlotRange → All, Joined → True]
```

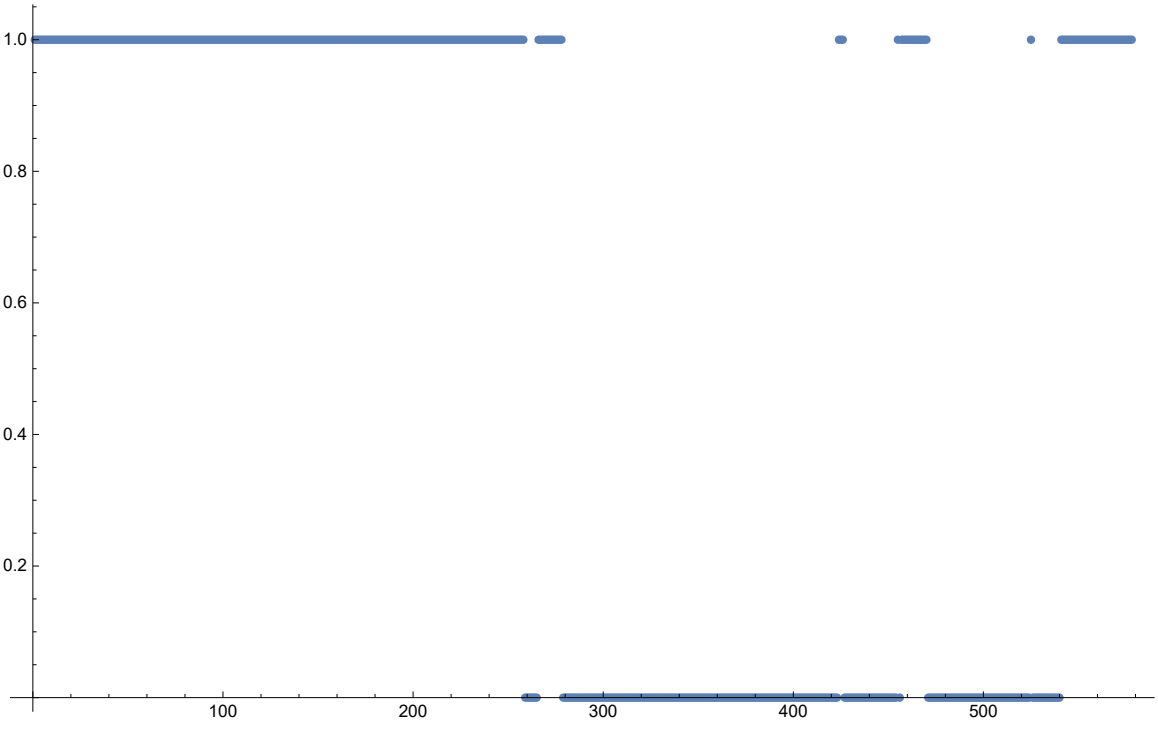
Out[18]=



Now look at the CloudFlag where a 1 indicates cloud, 0 indicates clear (Alexandrov et al., 2004).

```
In[19]:= ListPlot[CloudFlag]
```

Out[19]=

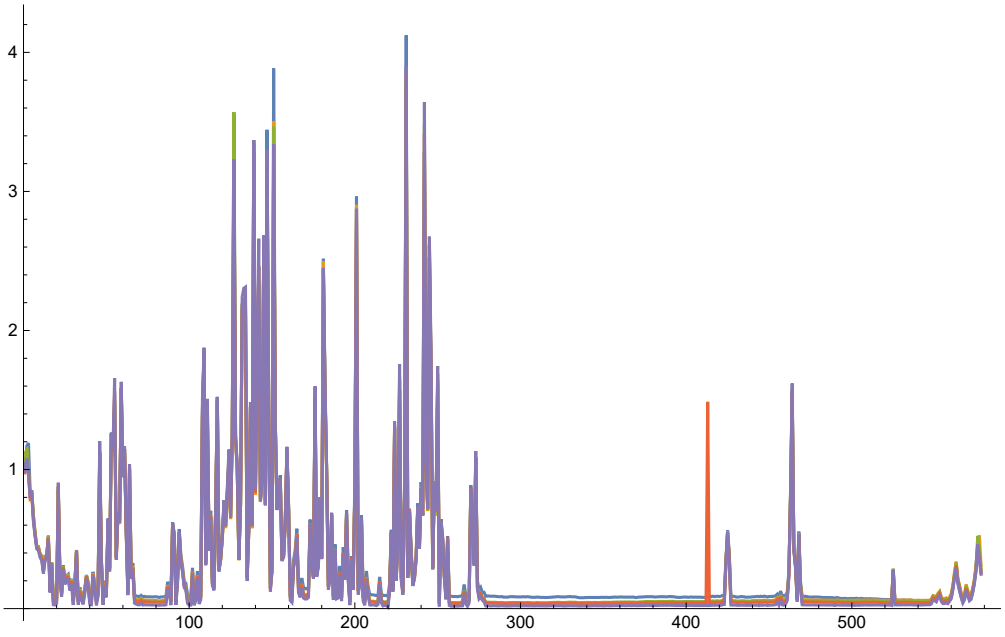


Look at the optical depths, OD, which is a combination of both clear and cloudy

measurements. You can see, by correlating the plots above and below that where the CloudFlag is 0, low ODs, consistent with aerosols exist.

In[20]:=
```
ListPlot[OD, Joined → True, PlotRange → All]
```

Out[20]=

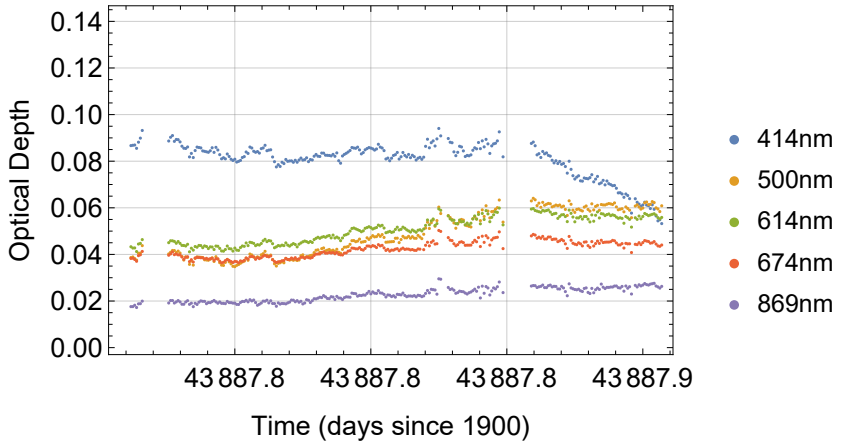


Now use the elements where CloudFlag=0 to select the clear cases thus obtaining just the aerosol optical depths and plot them up as a function of time. Your plot should look something like the one below. Note that the time field in the MFR files is days since 1900.

In[21]:=
```
AODs = Transpose[#] & /@ (Drop[#, 1] & /@ (Transpose[#] & /@
        (Select[#, #[[1]] == 0 &] & /@ (Transpose[{CloudFlag, MFRTime, #}] & /@ OD))));
```

In[22]:=
```
ListPlot[AODs, BaseStyle → {14, FontFamily → "Helvetica"},
 Frame → True, FrameLabel → {"Time (days since 1900)", "Optical Depth"},
 PlotLegends → {"414nm", "500nm", "614nm", "674nm", "869nm"}, GridLines → Automatic]
```

Out[22]=



Use a time function to determine what day the data were taken

In[23]:=
```
DateObject[{1900, 1, 0}] + 43 887 days
```

Out[23]= Day: Thu 27 Feb 2020

Competition!! Which language is more efficient for coding this up - Python, Matlab, R, Mathematica??

Below is the Mathematica code from above needed to read in the data, package into cloud - cleared AOD and plot up versus time. Do your best to reproduce something similar and THEN count the number of characters needed for your code.

Since we have so many Python coders ... who can write the most efficient Python example? Low total number of characters wins bragging rights!

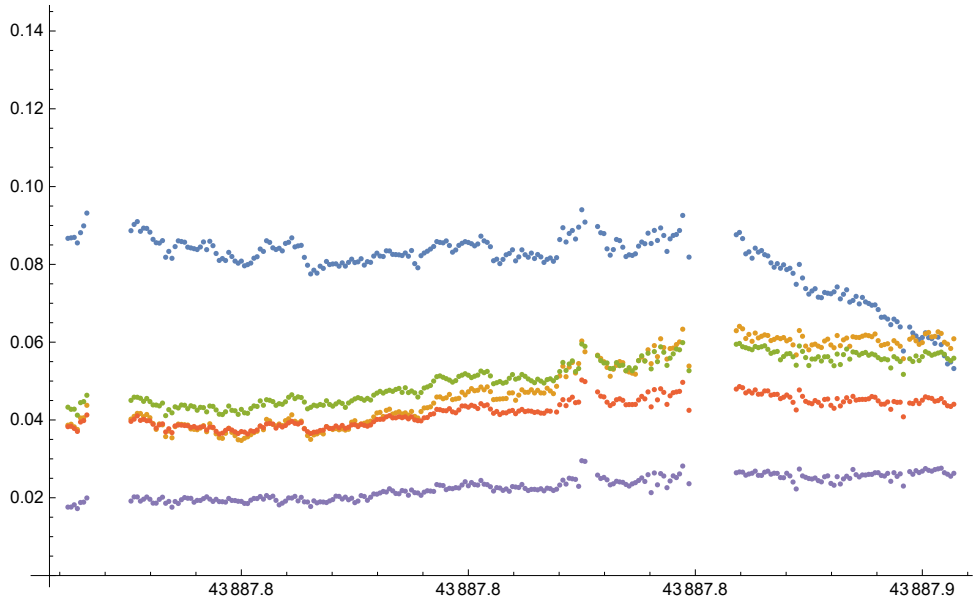
For comparison, the first version of code below requires 250 characters. The second version requires 127 characters.

```
{MFRTime, MFRVolts, OD, CloudFlag} = Import["Ret_677_20200227v0.9.h5",
   {"Datasets", {"MFRTime", "MFRVolts", "AOD", "CloudFlag"}}];
```

In[25]:=
```
AODs = Transpose[#] & /@ (Drop[#, 1] & /@ (Transpose[#] & /@
        (Select[#, #[[1]] == 0 &] & /@ (Transpose[{CloudFlag, MFRTime, #}] & /@ OD))));
```

In[26]:=
```
ListPlot[AODs]
```

Out[26]=



In[42]:=
```
119 + 117 + 14
```

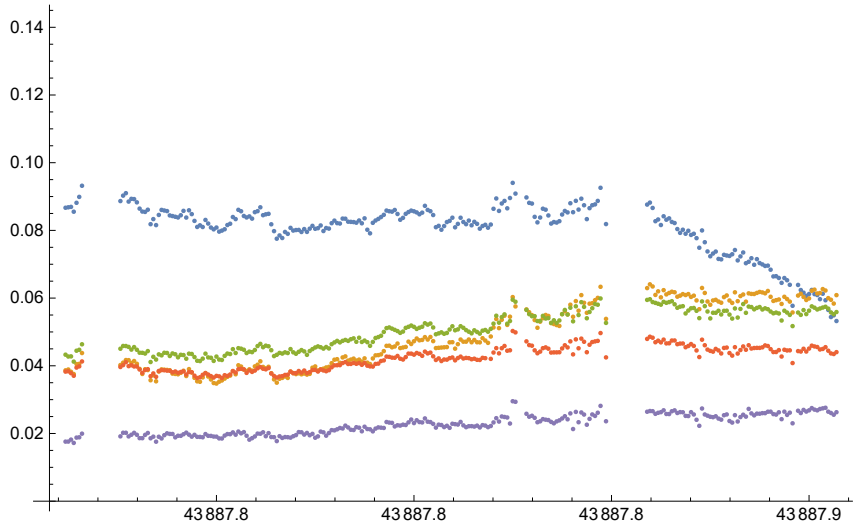
Out[42]= 250

Following some ideas of Iyasu from UTEP, this code avoids naming the datasets, etc and uses 127 characters

In[52]:=
```
i1 = Import["Ret_677_20200227v0.9.h5", "Data"];
```

In[53]:=
```
ListPlot[#ᵀ & /@ (Drop[#, 1] & /@
    (#ᵀ & /@ (Select[#, #[[1]] == 0 &] & /@ ({i1[[1]], i1[[8]], #}ᵀ & /@ i1[[12]]))))]
```

Out[53]=



In[54]:=
```
44 + 83
```

Out[54]= 127

Back to the main thread and the Ångström exponent ...

The Ångström exponent provides information about how particles scatter light as a function of wavelength. The exponent is strongly inversely correlated with particle size with smaller Ångström exponents being correlated with larger effective radius. The exponent, α, is derived from the relationship

$$\frac{\tau_a}{\tau_b} = \left(\frac{\lambda_a}{\lambda_b}\right)^{\alpha}$$

where τ_x is the aerosol optical thickness at wavelength λ_x.

In[•]:=
```
Clear[a, b, α]; Solve[τa/τb == (λa/λb)^-α, α]
```

... Solve: Inverse functions are being used by Solve, so some solutions may not be found; use Reduce for complete solution information.

Out[•]=
$$\left\{\left\{\alpha \to -\frac{\text{Log}\left[\frac{\tau_a}{\tau_b}\right]}{\text{Log}\left[\frac{\lambda_a}{\lambda_b}\right]}\right\}\right\}$$

Enrichment Excursion (i.e read this part to learn more about the conditions under which the Ångström formula is valid ...)

The use of the function **Solve** above makes assumptions about the kind of solution you want, but Mathematica also indicates that there are other conditions implied in the solution with the warning statement of "Inverse functions are being used ...". To find those other conditions, use the function **Reduce** as suggested

In[•]:=
```
Reduce[τa/τb == (λa/λb)^-α, α, Reals]
```

Out[•]=
$$\left(\lambda_b \neq 0 \&\& \tau_b \neq 0 \&\& \lambda_a == 0 \&\& \tau_a == 0 \&\& \alpha < 0\right) ||$$
$$\left(c_1 \in \mathbb{Z} \&\& \left(\tau_b < 0 || \tau_b > 0\right) \&\& \left(\left(\lambda_a < 0 \&\& \lambda_b > 0\right) || \left(\lambda_a > 0 \&\& \lambda_b < 0\right)\right) \&\&\right.$$
$$\left.\tau_a == \left(\frac{\lambda_a}{\lambda_b}\right)^{c_1} \tau_b \&\& \alpha == -c_1\right) || \left(c_1 \in \mathbb{Z} \&\& c_1 \geq 1 \&\& \left(\tau_b < 0 || \tau_b > 0\right) \&\&\right.$$
$$\left.\left(\left(\lambda_a < 0 \&\& \lambda_b > 0\right) || \left(\lambda_a > 0 \&\& \lambda_b < 0\right)\right) \&\& \tau_a == \left(\frac{\lambda_a}{\lambda_b}\right)^{c_1} \tau_b \&\& \alpha == -c_1\right) ||$$
$$\left(\lambda_a \neq 0 \&\& \tau_a \neq 0 \&\& \lambda_b == \lambda_a \&\& \tau_b == \tau_a\right) || \left(\text{Log}\left[\frac{\lambda_a}{\lambda_b}\right] \neq 0 \&\&\right.$$
$$\left(\left(\lambda_a < 0 \&\& \lambda_b < 0 \&\& \left(\left(\tau_a < 0 \&\& \tau_b < 0 \&\& \alpha == \frac{\text{Log}\left[\frac{\tau_b}{\tau_a}\right]}{\text{Log}\left[\frac{\lambda_a}{\lambda_b}\right]}\right) || \left(\tau_a > 0 \&\& \tau_b > 0 \&\& \alpha == \frac{\text{Log}\left[\frac{\tau_b}{\tau_a}\right]}{\text{Log}\left[\frac{\lambda_a}{\lambda_b}\right]}\right)\right)\right) ||\right.$$
$$\left.\left.\left(\lambda_a > 0 \&\& \lambda_b > 0 \&\& \left(\left(\tau_a < 0 \&\& \tau_b < 0 \&\& \alpha == \frac{\text{Log}\left[\frac{\tau_b}{\tau_a}\right]}{\text{Log}\left[\frac{\lambda_a}{\lambda_b}\right]}\right) || \left(\tau_a > 0 \&\& \tau_b > 0 \&\& \alpha == \frac{\text{Log}\left[\frac{\tau_b}{\tau_a}\right]}{\text{Log}\left[\frac{\lambda_a}{\lambda_b}\right]}\right)\right)\right)\right)\right)$$

What does this mean? Let' s parse the conditions above realizing that

Log - natural log
&& - logical “AND”
|| - logical “OR”
ϵ - is an element of
$\mathbb{Z}$ - the set of integers

Parsing the various options, the condition that pertains to the physical situation of our measurement is

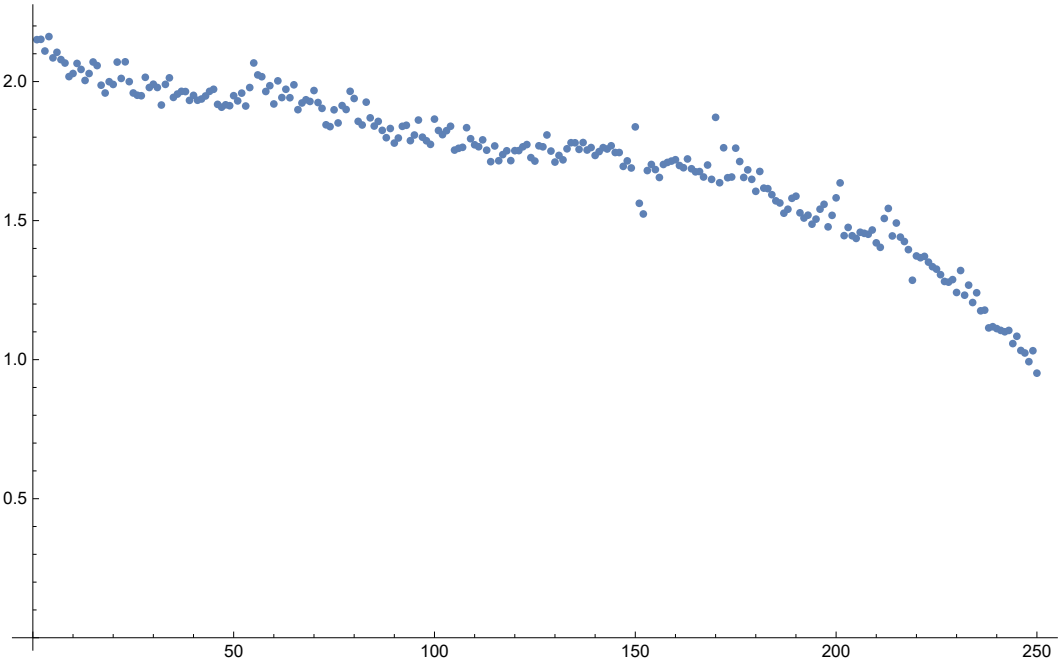
$$\text{Log}\left[\frac{\lambda_a}{\lambda_b}\right] \neq 0 \&\& \left(\lambda_a > 0 \&\& \lambda_b > 0 \&\& \left(\tau_a > 0 \&\& \tau_b > 0 \&\& \alpha == \frac{\text{Log}\left[\frac{\tau_b}{\tau_a}\right]}{\text{Log}\left[\frac{\lambda_a}{\lambda_b}\right]}\right)\right.$$

Returning to the main thread, evaluate now the Ångström exponent between 414 nm and 869 nm, realizing that all 5 AODs are packed in the variable called AODs with the values at 414 and 869 nm being the 1st and 5th of those, respectively.

In[•]:=
```
Ang414869 = -Log[Transpose[AODs[[1]]][[2]] / Transpose[AODs[[5]]][[2]]] / Log[414 / 869];
```

In[•]:= **ListPlot[Ang414869]**

Out[•]=



For comparison, figure 7 from Veselovskii et al., 2009 shows a correlation between Ångström exponent and particle effective radius retrieved from a multi-wavelength Raman lidar operating at NASA/GSFC in 2006. Based on this plot we would infer that the effective radius of the particles present on February 27, 2020 were increasing in effective radius during the measurement period from approximately 0.15 μm to 0.3 μm.

In[ ]:=

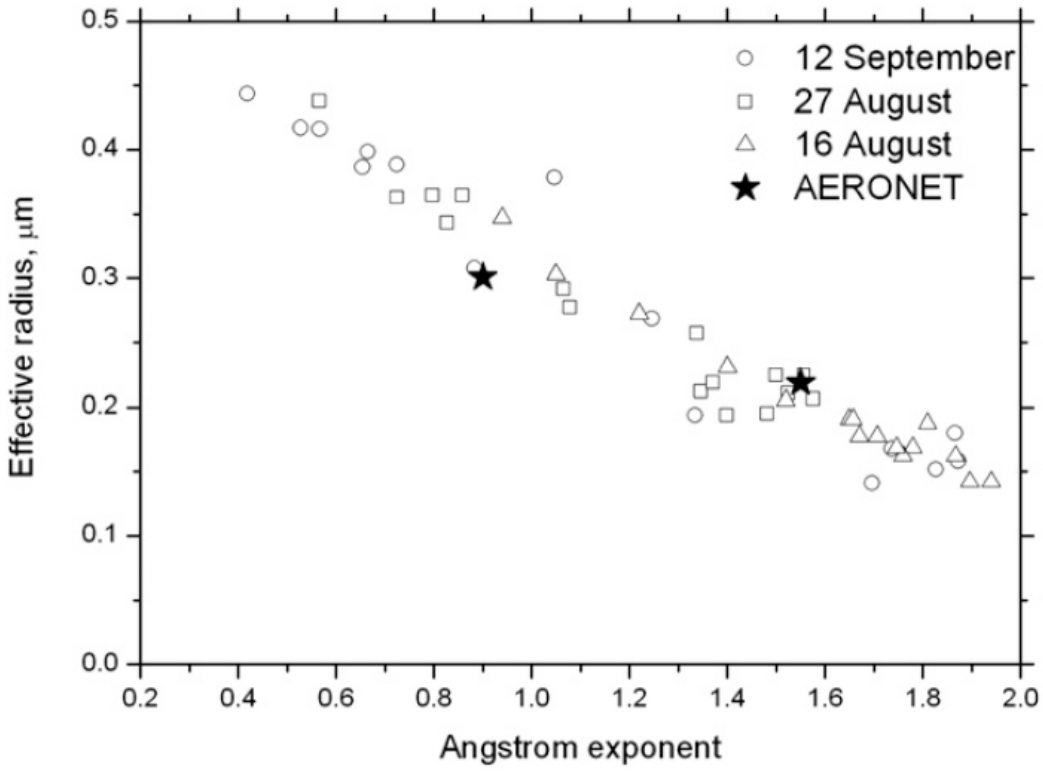


FIG. 7. Correlation between lidar-derived Ångström exponent and effective radius. Solid stars represent the results for AERONET on 16 Aug and 12 Sep.

Exercise 2 (ASCII and netCDF radiosonde data file import and manipulation).

Here we will work with datafiles from two different radiosonde packages launched on the same balloon. These "dual launches" have been occurring on an approximately weekly basis at HU Beltsville since 2016.

For these datafiles, the tasks in order are :
1) view the ascii files to discern the formats
2) import the data
3) compare some measurements between the RS92 and RS41
4) calculate water vapor mixing ratio using the Hyland Wexler, 1983 definition of saturation vapor pressure
5) compare some measurements between the RS92 and the GRUAN-processed RS92

We will work with sonde files acquired on January 23, 2020. On this day, there was a multi - sonde launch that included Vaisala RS92, Vaisala RS41 and Cryogenic Frostpoint Hygrometer (CFH). Those in the instrumentation track should learn more about these packages during the workshop.

To see what files are available at Beltsville from this launch, you can consult the spreadsheet that is automatically maintained that keeps track of the matchups. Go to the workshop website and switch to the “current measurements” tab and look for the set of “Dropbox Folders”. Click on the one for “Sondes” and then switch to the SondePlots folder. In this folder you can find the file “MultiLaunchSondes_1MostRecent.xls”. Consulting that spreadsheet indicates that the filenames of the data available from January 23, 2020 are:

1) RS41 file: **HUBV_RS41SGP_20200123_064641UT.mw41.dat**
2) RS92 file: **ALVICE_RS92SGP_20200123_064640UT.mw41.dat**
3) CFH file: **CFH_20200123_0646.bv064fle.dat**
4) GRUAN processed RS92: **BEL-RS-01_2_RS92-GDP_002_20200123T064600_1-003-001.nc**

The 3rd file is from a Cryogenic Frostpoint Hygrometer that is launched monthly at Beltsville as part of the international GRUAN effort. It is a very interesting instrument but we will ignore it for this exercise. The 4th of these files is a reprocessing of the RS92 data done by the GRUAN lead center in Lindenberg, Germany based on corrections developed by Miloshevich et al., 2006, 2009. We will now work on files 1, 2 and 4 above. These files are located in folders called RS41, RS92 etc in the same dropbox.

Unlike the hdf file used in Exercise 1, most of these files are in ASCII format. The good news is that you can just open up an ASCII file in a text editor and have a look at it. The bad news is that there is no standard way to format ASCII files so each file type requires some different coding to read it in. To get started with figuring out how to read an ASCII file, you have to look at the files. So let's have a look...

In[ ]:=
```
fileRS41 =
  "F:\\Dropbox\\Data\\Processed\\BVSondes\\RS41\\HUBV_RS41SGP_20200123_064641UT.mw41.dat";
fileRS92 =
  "F:\\Dropbox\\Data\\Processed\\BVSondes\\RS92\\ALVICE_RS92SGP_20200123_064640UT.mw41.
    dat";
fileGRUAN =
  "F:\\Dropbox\\Data\\Processed\\BVSondes\\GRUAN\\BEL-RS-01_2_RS92-GDP_002_20200123
    T064600_1-003-001.nc";
```

Have a look at the first part of the RS41 file which indicates that there are 40 header lines. You may prefer to do this in a separate program like notepad in windows. The header supplies useful information about the instrument, the launch location, etc. The descriptions of all the data columns are found after the line stating “Variable Unit”.

In[•]:=
```
TableForm[Take[Import[fileRS41], 50], TableSpacing → {0, 0.5}]
```

Out[•]//TableForm=

Generated	by	Rfunction: Get.mw41.edt.func2			
============>	Radisonde_info:				
RS_type:	RS41-SGP				
RS_config:	-32768				
RS_serialnum:	R3340183				
RS_freq:	403				
RS__windtype:	ccGPS				
============>	Station_info:				
Station:	HUBV_RS41SGP				
Latitude:	39.0563				
Longitude:	-76.8755				
Altitude:	52.3				
SW	version:	MW41	2.15.0		
Start	time:	2020-01-23	06:46:41		
============>	Variables	&	units	-	Vaisala I
NA_numeric	value:	-9999			
NA_string:	xx	or	NA		

Variable	Unit				
time	sec				
xx	NA				
Ta	K				
RH	%				
v(S->N)	m/s				
u(E->W)	m/s				
Height	m				
press	hPa				
Td	K				
MR	g/Kg				
DD	dgr				
FF	m/s				
Ascend_FLG	(0-N,1-Y)				
xx	NA				
xx	NA				
Lon	dgr				
Lat	dgr				
xx	NA				
xx	NA				
xx	NA				
============>	Data:				
0.	-9999.	268.37	85.	0. 0.	
0.81	-9999.	268.46	83.38	0.46 0.86	
1.81	-9999.	268.74	81.17	0.73 1.29	
2.81	-9999.	269.39	77.76	0.9 1.5	
3.81	-9999.	270.3	73.57	1.02 1.6	
4.81	-9999.	270.86	70.	1.14 1.66	
5.81	-9999.	271.23	67.49	1.26 1.72	
6.81	-9999.	271.7	65.76	1.39 1.79	
7.81	-9999.	272.1	64.18	1.53 1.86	
8.81	-9999.	272.37	62.6	1.67 1.92	

In order now to read in the desired variables we will drop the first 40 lines and package up the Time, Temperature, RH, Height, Pressure, Mixing Ratio. Fortunately, the RS92 has the same ASCII format so the read statements are similar.

In[•]:=
```
{timeRS41, tempKRS41, rhRS41, heightRS41, pressRS41, mixratRS41} =
  Transpose[Drop[Import[fileRS41], 40]][[{1, 3, 4, 7, 8, 10}]];
{timeRS92, tempKRS92, rhRS92, heightRS92, pressRS92, mixratRS92} =
  Transpose[Drop[Import[fileRS92], 40]][[{1, 3, 4, 7, 8, 10}]];
```

The reprocessed RS92 data that GRUAN produces is (thankfully) provided in a netCDF format. So you can interrogate this file like the hdf file in Exercise 1 and find out what the different variables are.

```
In[•]:= Import[fileGRUAN]

Out[•]= {time, press, temp, rh, wdir, wspeed, geopot, lon, lat, alt, u, v,
  FP, WVMR, asc, SWrad, u_SWrad, cor_temp, u_cor_temp, u_std_temp, u_temp,
  u_alt, u_press, res_rh, u_std_rh, cor_rh, u_cor_rh, u_rh, u_wdir, u_wspeed}
```

More information is available for each of the variables indicated above by interrogating the “Annotations” as follows. From this we see the units of time, pressure, temperature, etc as well as descriptions of the quantities.

```
In[•]:= TableForm[Import[fileGRUAN, "Annotations"], TableSpacing → {0, 0.5}]

Out[•]//TableForm=
```

standard_name → time	units → seconds since 2020-01-23T0€
standard_name → air_pressure	units → hPa
standard_name → air_temperature	units → K
standard_name → relative_humidity	units → 1
standard_name → wind_from_direction	units → degree
standard_name → wind_speed	units → m s-1
standard_name → geopotential_height	units → m
standard_name → longitude	units → degree_east
standard_name → latitude	units → degree_north
standard_name → altitude	units → m
standard_name → eastward_wind	units → m s-1
standard_name → northward_wind	units → m s-1
units → K	long_name → Frostpoint
standard_name → Water_vapor_mixing_ratio	units → 1
units → m s-1	long_name → Ascent/Descent Speed
standard_name → short_wave_radiation	units → W m-2
standard_name → short_wave_radiation standard_error	units → W m-2
standard_name → air_temperature correction	units → K
standard_name → air_temperature correlated_uncertainty	units → K
standard_name → air_temperature standard_deviation	units → K
standard_name → air_temperature standard_error	units → K
standard_name → altitude standard_error	units → m
standard_name → air_pressure standard_error	units → hPa
standard_name → relative_humidity resolution	units → s
standard_name → relative_humidity standard_deviation	units → 1
standard_name → relative_humidity correction	units → 1
standard_name → relative_humidity correlated_uncertainty	units → 1
standard_name → relative_humidity standard_error	units → 1
standard_name → wind_from_direction standard_error	units → degree
standard_name → wind_speed standard_error	units → m s-1

Now read the same parameters from the GRUAN processed file as were read from the original RS92 file

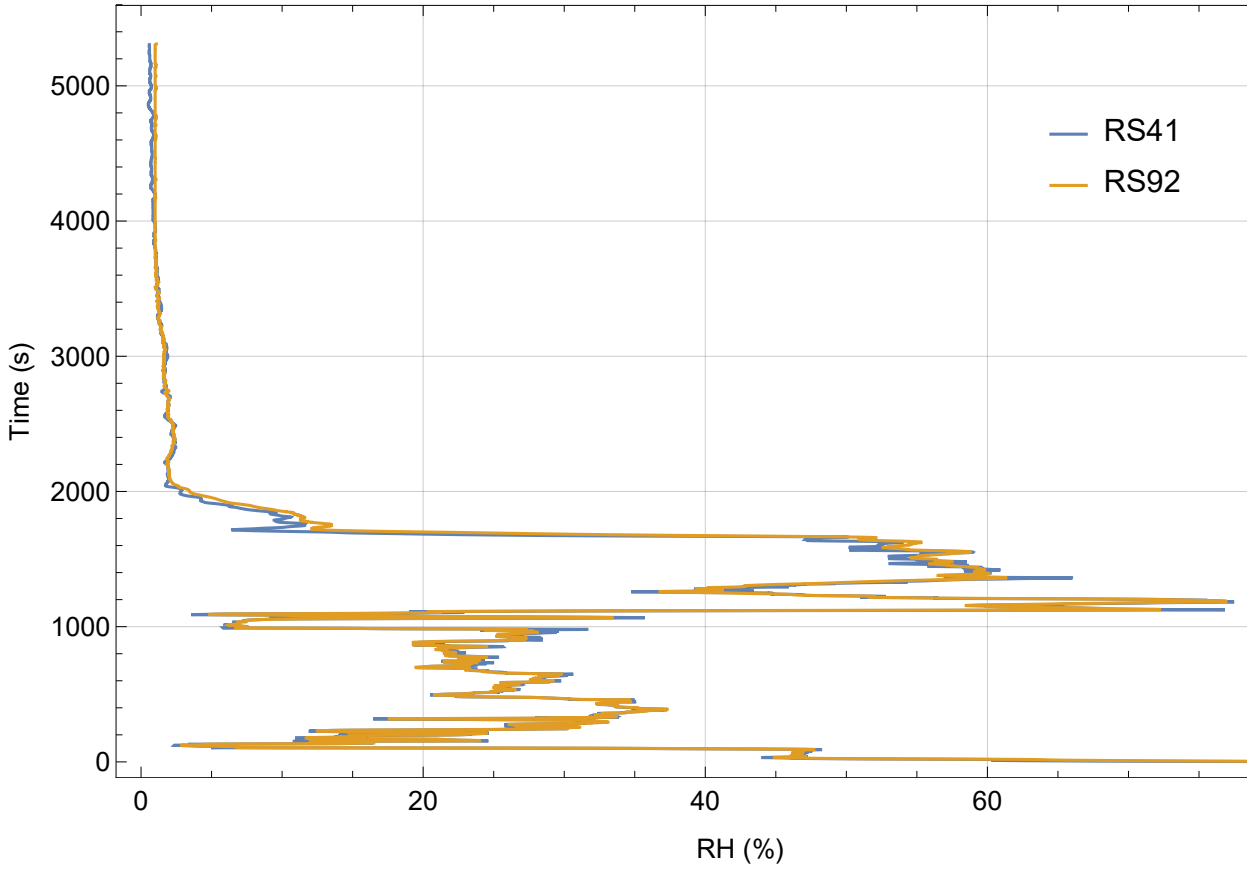
```
In[•]:= {timeRS92G, tempKRS92G, rhRS92G, heightRS92G, pressRS92G, mixratRS92G} =
    Import[fileGRUAN, {"Datasets", {"time", "temp", "rh", "geopot", "press", "WVMR"}}];
```

So we have the data from all three sonde files and can start with some comparisons. The best independent variable to use for a multi-sonde comparison where the instruments are on the same balloon is time (instead of height). The reason for this is that clocks are extremely precise whereas height

calculations can be subject to pressure or GPS errors. So to compare the RH measurements of the RS41 and RS92 that were on the same balloon, we will do so as a function of time.

In[ ]:=
```
ListPlot[{Thread[{rhRS41, timeRS41}], Thread[{rhRS92, timeRS92}]},
 BaseStyle → {14, FontFamily → "Helvetica"}, Frame -> True,
 FrameLabel → {"RH (%)", "Time (s)"}, Joined → True,
 GridLines → Automatic, PlotLegends → Placed[{"RS41", "RS92"}, {0.8, 0.8}]]
```

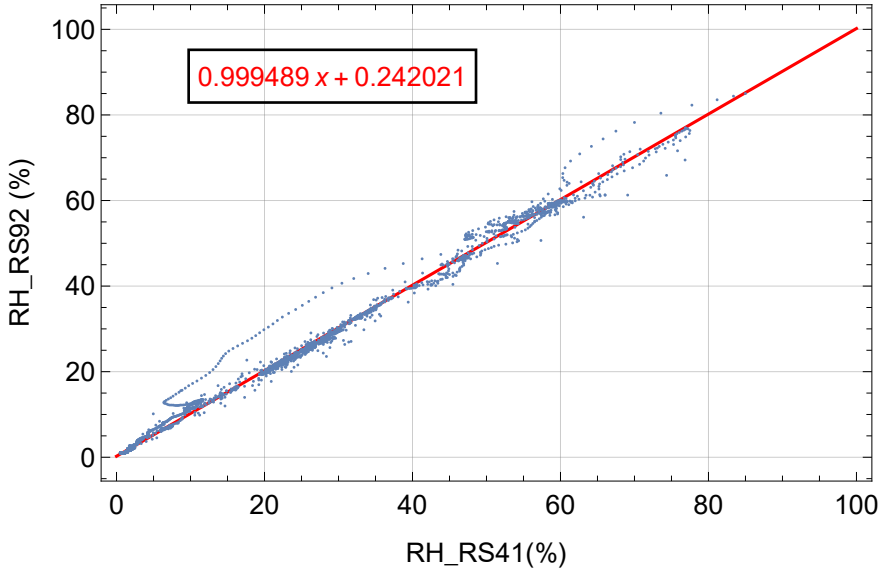
Out[ ]=



The humidity sensor of the RS41 has much faster response than that of the RS92. Above you can see that the RS41 shows significant departures from the RS92 in regions of rapidly changing values in RH due to this faster response. Now create a scatter plot to look at how the RH measurements compare over the range of RH. To form the ordered pairs, make a linear interpolation function for the RS92 profile and then evaluate at the timebase of the RS41. You can use these ordered pairs to determine the linear best fit function. The significant departures from the best fit line shown below are no doubt due to the regions of discrepancy shown above where RH is changing rapidly.

In[ ]:=
```
lm = LinearModelFit[pairsRH = Thread[{rhRS41,
      Interpolation[Thread[{timeRS92, rhRS92}], InterpolationOrder → 1][timeRS41]}], x, x];
Show[Plot[lm[x], {x, 0, 100}, PlotStyle → Red], ListPlot[pairsRH], Frame → True,
 GridLines → Automatic, BaseStyle → {14, FontFamily → "Helvetica"},
 Frame -> True, FrameLabel → {"RH_RS41(%)", "RH_RS92 (%)"},
 Epilog → Inset[Framed[Style[Normal[lm], Red]], Scaled[{0.3, 0.85}]]]
```

Out[ ]=



Consider now the calculation of water vapor mixing ratio. This is a derived quantity from the radiosonde data since the sonde measures RH, temperature and pressure and not mixing ratio. In their calibration of the radiosondes, Vaisala uses the Wexler, 1976 and Hyland - Wexler, 1983 formulations of the saturation vapor pressure over water and ice, respectively (Miloshevich et al., 2006). So it is best to use these formulations when converting from RH to Mixing Ratio. Surprisingly, the data files for radiosondes sometimes use other

formulations that can lead to significant differences.

Now for some equations and formulas...to convert from RH to Mixing Ratio, use the following:

Water Vapor Mixing Ratio (g/kg) = 10 RH $\frac{MW_w}{MW_D} \frac{e_s}{P - \frac{RH}{100} e_s}$

where,

RH is in %
$\frac{MW_w}{MW_D}$ is the ratio of molecular weights of water vapor and dry air and equal to approximately 0.62197
e_s is the saturation vapor pressure of water vapor
P is pressure in hPa or mbarr

To follow the recommendations of Elliott and Gaffen, 1991 and the procedures used by Vaisala in their calibration (Miloshevich et al., 2006), we use the Wexler, 1976 formulation of the saturation vapor pressure which is one of the choices in the function SatVaporPressLiq defined below.

```
In[•]:= GetMixRatioWater[{TempC_, PressMB_, RH_}] :=
         10 RH SaturationMixingRatioWater2[{TempC, PressMB, RH}] (* output in g/kg *)

In[•]:= SaturationMixingRatioWater2[{TempC_, PressMB_, RHLiqPercent_}] := Module[{SatVapPress},
          SatVapPress = SatVaporPressLiq[TempC, "Wexler"];
          0.62197 SatVapPress / (PressMB - RHLiqPercent/100 SatVapPress)
          (* output in units of kg/kg *)
         ]
```

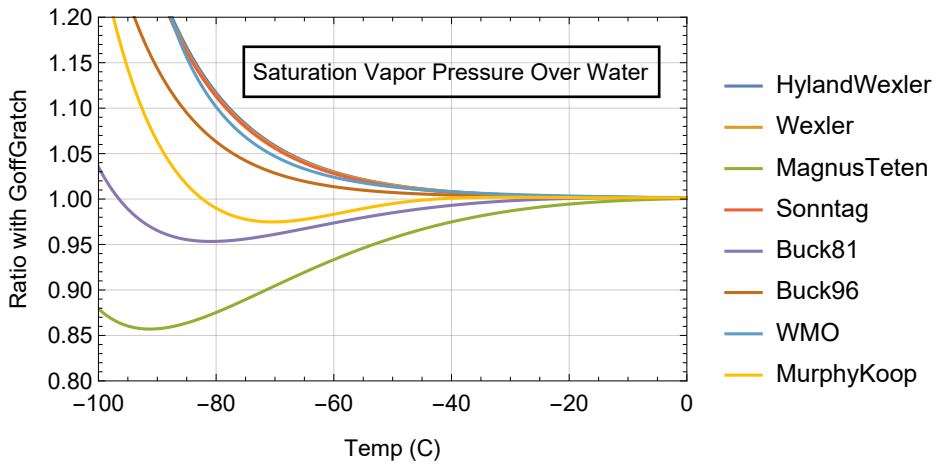
```
In[•]:= SatVaporPressLiq[TempC_, Choice_] := Module[{TempK},

  (* output in hPa i.e. mb *)

  TempK = TempC + 273.15;
  Switch[Choice,
   "GoffGratch", 10^(-7.90298 (373.16 / TempK - 1) +
      5.02808 Log[10, 373.16 / TempK] - 1.3816 × 10^-7 (10^11.344 (1-TempK/373.16) - 1) +
      8.1328 × 10^-3 (10^-3.49149 (373.16/TempK -1) - 1) + Log[10, 1013.246]),
   "HylandWexler", Exp[(-0.58002206 × 10^4 / TempK + 0.13914993 × 10^1 -
       0.48640239 × 10^-1 TempK + 0.41764768 × 10^-4 TempK^2 -
       0.14452093 × 10^-7 TempK^3 + 0.65459673 × 10^1 Log[TempK])] 10^-2,
   "Wexler", Exp[-2.9912729 × 10^3 TempK^-2 - 6.0170128 × 10^3 TempK^-1 + 1.887643854 × 10^1 -
      2.8354721 × 10^-2 TempK + 1.7838301 × 10^-5 TempK^2 - 8.4150417 × 10^-10 TempK^3 +
      4.4412543 × 10^-13 TempK^4 + 2.858487 Log[TempK]] 10. /1000.,
   "MagnusTeten", 10^((7.5 TempC) / (TempC + 237.3) + 0.7858),
   "Sonntag", Exp[-6096.9385 / TempK + 16.635794 -
     2.711193 × 10^-2 TempK + 1.673952 × 10^-5 TempK^2 + 2.433502 Log[TempK]],
   "Buck81", 6.1121 Exp[17.502 TempC / (240.97 + TempC)],
   "Buck96", 6.1121 Exp[(18.678 - TempC / 234.5) TempC / (257.14 + TempC)],
   "WMO", 10^(10.79574 (1 - 273.15 / TempK)
      - 5.02800 Log[10, TempK / 273.15] + 1.50475 × 10^-4 (1 - 10^-8.2969 (TempK/273.15-1)) +
      0.42873 × 10^-3 (10^-4.769955 (1-273.15/TempK) - 1) + 0.78614),
   "MurphyKoop", Exp[(54.842763 - 6763.22 / TempK - 4.21 Log[TempK] +
       0.000367 TempK + Tanh[0.0415 (TempK - 218.8)]
        (53.878 - 1331.22 / TempK - 9.44523 Log[TempK] + 0.014025 TempK))] 10^-2
  ]
 ]
```

```
In[•]:= SatVaporPressIce[TempC_, Choice_] := Module[{TempK},

  (* output in hPa i.e. mb *)

  TempK = TempC + 273.15;
  Switch[Choice,
   "GoffGratch", 10^(-9.09718 (273.15 / TempK - 1) -
      3.56654 Log[10, 273.15 / TempK] + 0.876793 (1 - TempK / 273.15) + Log[10, 6.1071]),
   "HylandWexler", Exp[(-0.56745359 × 10^4 / TempK + 0.63925247 × 10^1 -
       0.96778430 × 10^-2 TempK + 0.62215701 × 10^-6 TempK^2 + 0.20747825 × 10^-8 TempK^3 -
       0.94840240 × 10^-12 TempK^4 + 0.41635019 × 10^1 Log[TempK])] 10^-2,
   "MagnusTeten", 10^((9.5 TempC) / (TempC + 265.5) + 0.7858),
   "Buck81", 6.1115 Exp[22.452 TempC / (272.55 + TempC)],
   "Buck96", 6.1115 Exp[(23.036 - TempC / 333.7) TempC / (279.82 + TempC)],
   "WMO", 10^(-9.09685 (273.15 / TempK - 1)
      - 3.56654 Log[10, 273.15 / TempK] + 0.87682 (1 - TempK / 273.15) + 0.78614),
   "MurphyKoop",
   Exp[9.550426 - 5723.265 / TempK + 3.53068 Log[TempK] - 0.00728332 TempK] 10^-2
  ]
 ]
```

Enrichment Excursion: Before moving on, let' s have a look at all those different formulations of saturation vapor pressure over both water and ice. Below we plot the ratio of each of the liquid formulations with that of the GoffGratch equation.

```
In[•]:= TheseTemps = Table[x, {x, -100, 50}];
 ggVals = SatVaporPressLiq[#, "GoffGratch"] & /@ TheseTemps;
 ListPlot[
  {Thread[{TheseTemps, (SatVaporPressLiq[#, "HylandWexler"] & /@ TheseTemps) / ggVals}],
   Thread[{TheseTemps, (SatVaporPressLiq[#, "Wexler"] & /@ TheseTemps) / ggVals}],
   Thread[{TheseTemps, (SatVaporPressLiq[#, "MagnusTeten"] & /@ TheseTemps) / ggVals}],
   Thread[{TheseTemps, (SatVaporPressLiq[#, "Sonntag"] & /@ TheseTemps) / ggVals}],
   Thread[{TheseTemps, (SatVaporPressLiq[#, "Buck81"] & /@ TheseTemps) / ggVals}],
   Thread[{TheseTemps, (SatVaporPressLiq[#, "Buck96"] & /@ TheseTemps) / ggVals}],
   Thread[{TheseTemps, (SatVaporPressLiq[#, "WMO"] & /@ TheseTemps) / ggVals}],
   Thread[{TheseTemps, (SatVaporPressLiq[#, "MurphyKoop"] & /@ TheseTemps) / ggVals}]},
  Frame → True, GridLines → Automatic, PlotLegends → {"HylandWexler", "Wexler",
    "MagnusTeten", "Sonntag", "Buck81", "Buck96", "WMO", "MurphyKoop"},
  PlotRange → {{-100, 0}, {0.8, 1.2}}, Joined → True,
  BaseStyle → {12, FontFamily → "Helvetica"},
  FrameLabel → {"Temp (C)", "Ratio with GoffGratch"},
  Epilog → Inset[Framed[Style["Saturation Vapor Pressure Over Water", "Helvetica"]],
    Scaled[{0.6, 0.85}]]]
```

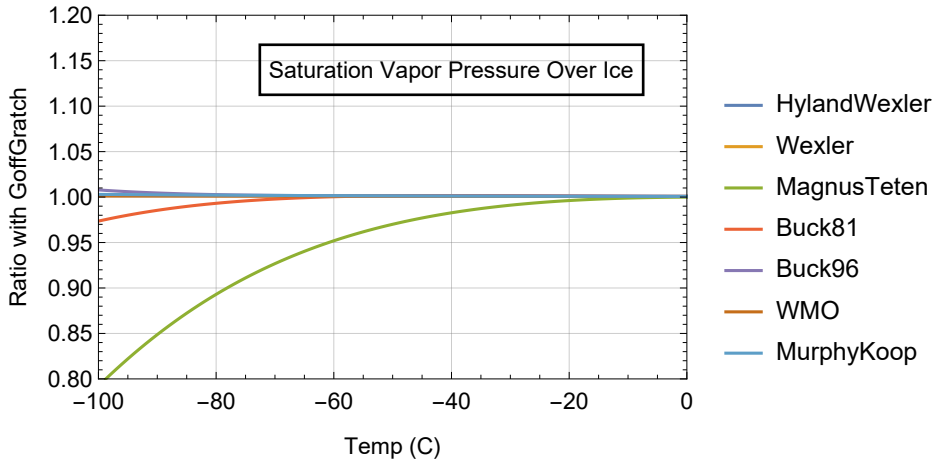
Out[ ]=



Note how the curves above are in very good agreement near the freezing point of water but at lower temperatures diverge very noticeably. A significant reason for this is that the homogeneous nucleation point of water is about - 37 C so the concept of a saturation vapor pressure over liquid water at such cold temperatures is not really physical and thus cannot be measured in the lab. Nonetheless, the convention for reporting RH from radiosondes is to use RH over water even at these very cold temperatures. By contrast, look at various formulations of RH over ice shown below.

In[ ]:=
```
ggValsIce = SatVaporPressIce[#, "GoffGratch"] & /@ TheseTemps;
ListPlot[
 {Thread[{TheseTemps, (SatVaporPressIce[#, "HylandWexler"] & /@ TheseTemps) / ggValsIce}],
  Thread[{TheseTemps, (SatVaporPressIce[#, "Wexler"] & /@ TheseTemps) / ggValsIce}],
  Thread[{TheseTemps, (SatVaporPressIce[#, "MagnusTeten"] & /@ TheseTemps) / ggValsIce}],
  Thread[{TheseTemps, (SatVaporPressIce[#, "Buck81"] & /@ TheseTemps) / ggValsIce}],
  Thread[{TheseTemps, (SatVaporPressIce[#, "Buck96"] & /@ TheseTemps) / ggValsIce}],
  Thread[{TheseTemps, (SatVaporPressIce[#, "WMO"] & /@ TheseTemps) / ggValsIce}],
  Thread[{TheseTemps, (SatVaporPressIce[#, "MurphyKoop"] & /@ TheseTemps) / ggValsIce}]},
 Frame → True, GridLines → Automatic, PlotLegends →
  {"HylandWexler", "Wexler", "MagnusTeten", "Buck81", "Buck96", "WMO", "MurphyKoop"},
 PlotRange → {{-100, 0}, {0.8, 1.2}}, Joined → True,
 BaseStyle → {12, FontFamily → "Helvetica"},
 FrameLabel → {"Temp (C)", "Ratio with GoffGratch"},
 Epilog → Inset[Framed[Style["Saturation Vapor Pressure Over Ice", "Helvetica"]],
   Scaled[{0.6, 0.85}]]]
```

Out[ ]=

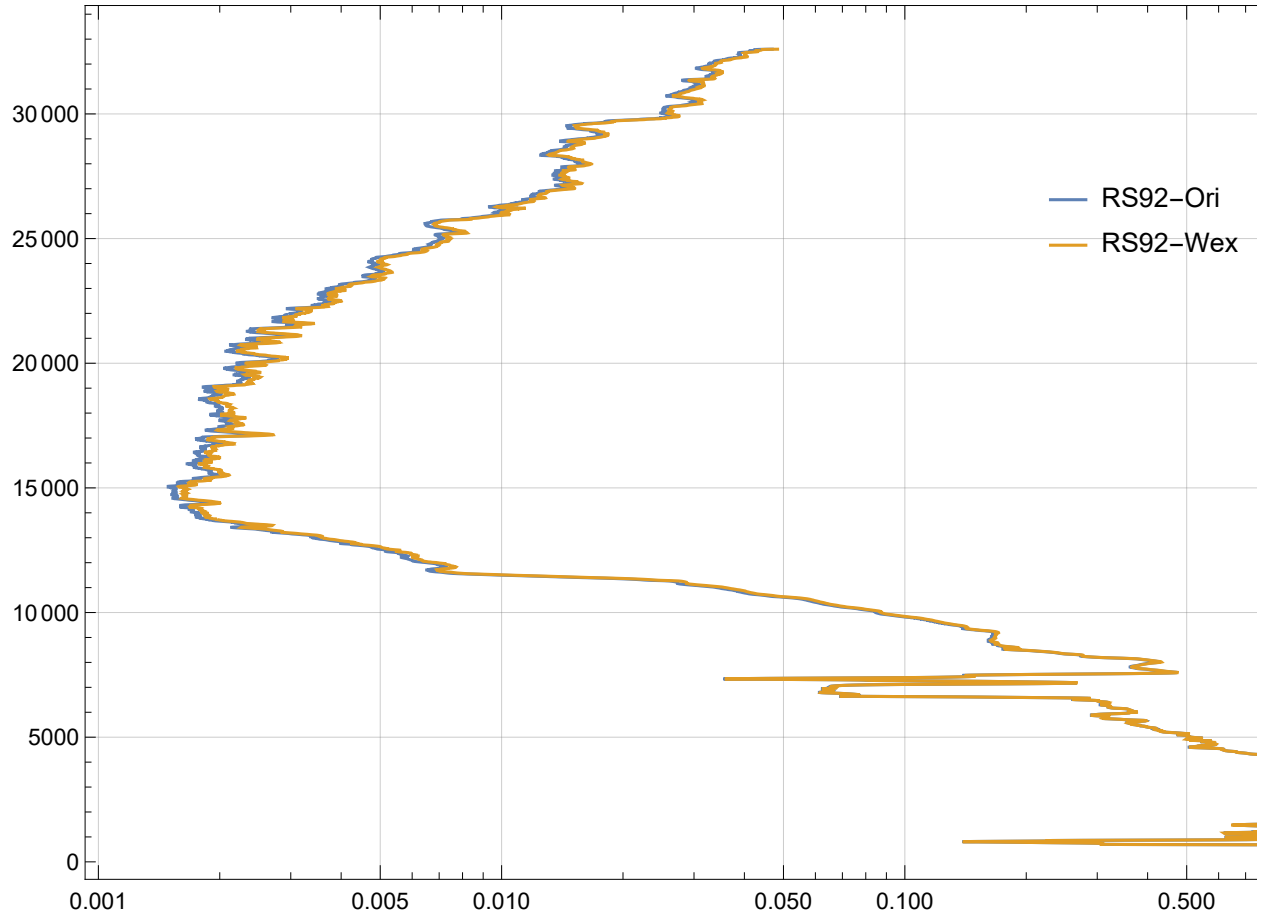

Saturation Vapor Pressure Over Ice
Ratio with GoffGratch
Temp (C)
1.20
1.15
1.10
1.05
1.00
0.95
0.90
0.85
0.80
−100
−80
−60
−40
−20
0
HylandWexler
Wexler
MagnusTeten
Buck81
Buck96
WMO
MurphyKoop

Because of generally much better agreement of these formulations of vapor pressure over ice versus temperatures below 0 (except Magnus Teeten), which result from actual laboratory measurements, some, including Miloshevich et al., 2006, have recommended that radiosonde datafiles also report RH over ice in addition to RH over water. Such reporting could eliminate most of the discrepancies due to the choice of vapor pressure formulation, however those suggestions have not yet been adopted by the community.

Now let' s calculate the mixing ratio from the T, P, RH measured by the RS92 sonde using the Wexler, 1976 formulation of the vapor pressure and compare with the mixing ratio reported in the RS92 sonde data file.

```
In[ ]:= ListLogLinearPlot[{Thread[{mixratRS92, heightRS92}],
          Thread[{wexRS92 = (GetMixRatioWater[{#[[1]] - 273.15, #[[2]], #[[3]]}] & /@
                Thread[{tempKRS92, pressRS92, rhRS92}]), heightRS92}]},
        PlotRange → All, Frame → True, BaseStyle → {12, FontFamily → "Helvetica"},
        Joined → True, GridLines → Automatic,
        PlotLegends → Placed[{"RS92-Ori", "RS92-Wex"}, Scaled[{0.75, 0.75}]]]
```

Out[ ]=


30 000
25 000
20 000
15 000
10 000
5000
0
0.001
0.005
0.010
0.050
0.100
0.500
RS92−Ori
RS92−Wex

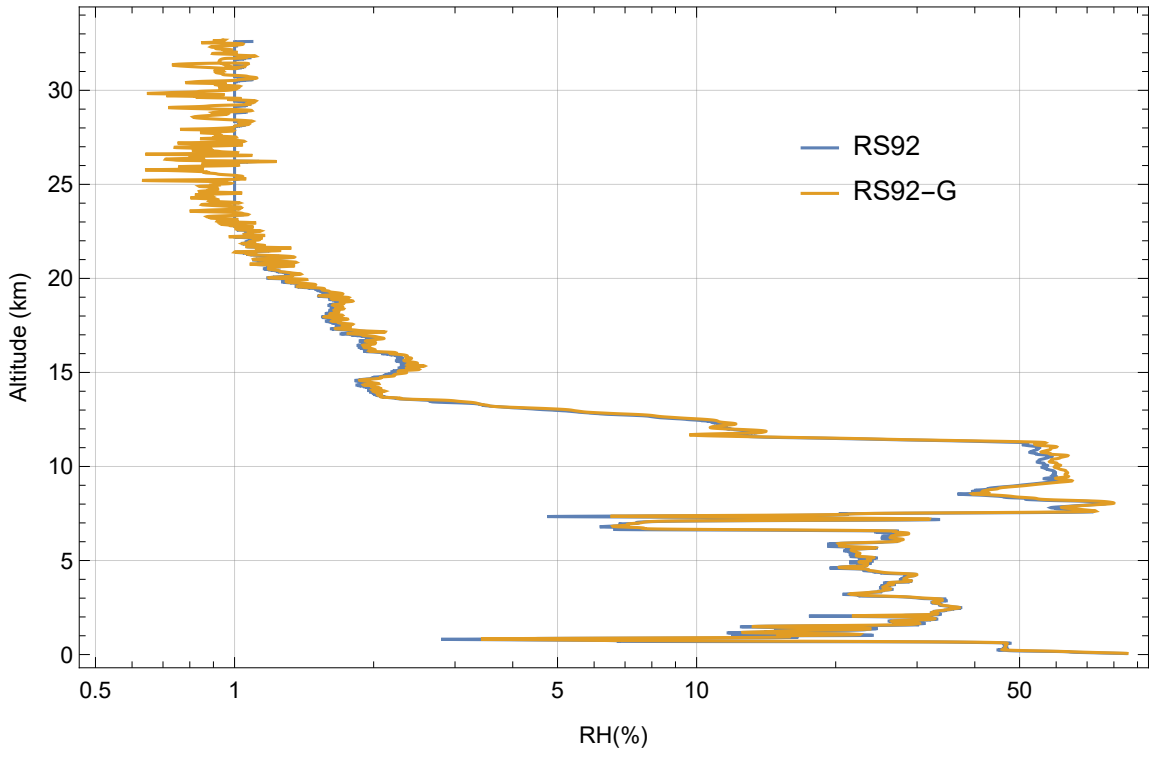
Discrepancies exist and are most evident around the very dry, cold tropopause at ~15 km indicating that Vaisala may have used a different formulation either for vapor pressure or for the conversion from RH to mixing ratio.

Exercise for the student!! Play around with the different vapor pressure formulations and see if you can resolve the discrepancies shown in the plot above. You may want to plot the data up differently to illustrate the discrepancy between the two mixing ratio calculations.

Finally let' s compare the normal RS92 file and the one generated using the GRUAN data processing. The largest differences are in the water vapor field (although other corrections are applied). So let' s compare the RH fields.

In[ ]:=
```
ListLogLinearPlot[
 {Thread[{rhRS92, heightRS92 / 1000}], Thread[{100 rhRS92G, heightRS92G / 1000}]},
 PlotRange → All, Frame → True, FrameLabel → {"RH(%)", "Altitude (km)"},
 BaseStyle → {12, FontFamily → "Helvetica"}, Joined → True, GridLines → Automatic,
 PlotLegends → Placed[{"RS92", "RS92-G"}, Scaled[{0.75, 0.75}]]]
```

Out[ ]=



The correction for the known dry bias in the RS92 is evident from a comparison of GRUAN - processed and original radiosonde data and can be seen perhaps

most noticeably at an altitude of approximately 10 km.

Exercise for the student! Present the data in different ways to quantify the magnitude of the dry bias correction. How much is the correction a function of RH? How much is it a function of temperature?

References

- Bolton, D., The computation of equivalent potential temperature, Monthly Weather Review, 108, 1046-1053, 1980.
- Buck, A. L., New equations for computing vapor pressure and enhancement factor, J. Appl. Meteorol., 20, 1527-1532, 1981.
- Buck Research Manuals, 1996
- Detwiler, A., Extrapolation of the Goff-Gratch formula for vapor pressure over liquid water at temperatures below 0°C, J. Appl. Meteorol., 22, 503, 1983.
- Elliott,W. P., and D. J. Gaffen (1991), On the utility of radiosonde humidity archives for climate studies, Bull. Am. Meteorol. Soc., 72, 1507–1520.
- Elliott, W. P. and D. J. Gaffen, Effects of conversion algorithms on reported upper air dewpoint depressions, Bull. Am. Meteorol. Soc., 74, 1323-1325, 1993.
- Fukuta, N. and C. M. Gramada, Vapor pressure measurement of supercooled water, J. Atmos. Sci., 60, 1871-1875, 2003.
- Gibbins, C. J., A survey and comparison of relationships for the determination of the saturation vapour pressure over plane surfaces of pure water and of pure ice, Annales Geophys., 8, 859-886, 1990.
- Goff, J. A., and S. Gratch, Low-pressure properties of water from -160 to 212 F, in Transactions of the American society of heating and ventilating engineers, pp 95-122, presented at the 52nd annual meeting of the American society of heating and ventilating engineers, New York, 1946.
- Goff, J. A. Saturation pressure of water on the new Kelvin temperature scale, Transactions of the American society of heating and ventilating engineers, pp 347-354, presented at the semi-annual meeting of the American society of heating and ventilating engineers, Murray Bay, Que. Canada, 1957.
- Hardy, B., 1998, ITS-90 Formulations for Vapor Pressure, Frostpoint Temperature, Dewpoint Temperature, and Enhancement Factors in the Range –100 to +100 °C, The Proceedings of the Third International Symposium on Humidity & Moisture, London, England
- Hyland, R. W., and A. Wexler, Formulations for the thermodynamic properties of the saturated phases of H2O from 173.15 K to 473.15 K, ASHRAE Trans., 2A, 500– 519 (1983).

- Marti, J. and K Mauersberger, A survey and new measurements of ice vapor pressure at temperatures between 170 and 250 K, GRL 20, 363-366, 1993
- Miloshevich, L.M., H.Voemel, D. Whiteman, B. Lesht, F.J. Schmidlin, and F. Russo (2006), Absolute accuracy of water vapor measurements from six operational radiosonde types launched during AWEX - G and implications for AIRS validation, J.Geophys.Res., 111, doi : 10.1029/2005 JD006083 (2006)
- Miloshevich LM, Vomel H, Whiteman DN, T. Leblanc, Accuracy assessment and correction of Vaisala RS92 radiosonde water vapor measurements, J. Geophys. Res, Vol. 114, D11305 (2009)
- Murphy, D. M. and T. Koop, Review of the vapour pressures of ice and supercooled water for atmospheric applications, Quart. J. Royal Met. Soc, 131, 1539-1565, 2005.
- Murray, F. W., On the computation of saturation vapor pressure, J. Appl. Meteorol., 6, 203-204, 1967.
- Wexler, A. (1976), Vapor pressure formulation for water in range 0 to 100°C: A revision, ., Res. Natl. Bur. Stand. U.S., Sect. A, 80, 775– 785.
- Smithsonian Met. Tables, 5th ed., pp. 350, 1984.
- Sonntag, D., Advancements in the field of hygrometry, Meteorol. Z., N. F., 3, 51-66, 1994.
- Veselovskii, I, D. N. Whiteman, A. Kolgotin, E. Andrews, M. Korenskii, Retrieval of Aerosol Physical Properties Under Varying Relative Humidity Conditions, J. Atmos. Ocean. Tech., (2009)
- Wagner W. and A. Pruß, The IAPWS formulation 1995 for the thermodynamic properties of ordinary water substance for general and scientific use, J. Phys. Chem. Ref. Data, 31, 387-535, 2002.
- Wexler, A., Vapor Pressure Formulation for Water in Range 0 to 100°C. A Revision, Journal of Research of the National Bureau of Standards, 80A, 775-785, 1976.
- World Meteorological Organization, Technical Regulations, Basic Documents No. 2, Volume I - General meteorological standards and recommended practices, Appendix A, WMO-No. 49, Geneva 2011, updated 2012.
- World Meteorological Organization, Guide to Meteorological Instruments and Methods of Observation, Appendix 4B, WMO-No. 8 (CIMO Guide), Geneva 2008.