## Lab 6 - Plotting weather station data

### Introduction

**Objective** - Learn how to pre-process data and create simple data visualizations of common atmospheric variables using data collected from your recent weather station deployment. You may use whatever plotting language you choose as long as all the required graphs are produced and look professional. Part of this assignment is learning to properly comment your code. You may *not* use MS Excel or any other spreadsheet program to complete this assignment. You will be required to submit your computer code used to generate the graphics *and* the data file you used. Instructions for this will be at the end of this tutorial.

**Overview of the assignment**

1. Pre-process raw data file

2. Parse file and re-format - READ data file into Matlab (or other plotting language) - Convert raw data file into a .MAT or NetCDF file.

3. Create meaningful graphs - READ post-processed data file into plotting language - Run data retrievals - Plot the data

You should read the short summary of [General Code Practices](general-code-practices.md) before you go much farther. Your code will not only be graded on its ability to function properly, but is it easy to read and follows general conventions.

### Pre-process raw data file

Ideally in the real world you would want to automate this process by creating a computer script to do this work for you as often during a field campaign you often collect more data than you know what to do with. Also by automating this process you reduce the human errors such as mis-treating the data (“treating” meaning preparing the data). You will see in the “Cupsonde” lab, we have provided you with a Perl script to automate the pre-processing of the data. For this lab you can do this quickly for your single data file using any text editor of your choice. Below is an example of how to do this using a regular text editor and *Vi*.

Using a standard text editor simple do the following: - Open your data file - Search and replace the following: - **“NAN”** with **-9999** - Save and close the file

With *Vi* you can do the following from the command line

vi CFEE2.dat

You will need to type : to tell *Vi* to enter edit mode. Now you should type the following at the prompt

%s/"NAN"/-9999/

This will replace all the **“NAN”** with **-9999**. Then press escape. To exit and save your file in *Vi*, hold down the shift key type zz.

Regardless if you used *Vi* or not, please read http://www.linfo.org/vi/search.html and explain the syntax of the search command used above as part of your lab report.

### Read data into Matlab

Using the following code, read in your pre-processed data file and save file the file as a MAT file. For other languages please store as a NetCDF/HDF file. If you are truly brave and do this assignment in C/Fortran & GNUPLot, you can use whatever text file format you choose.

Before we get too far ahead of ourselves lets think about the steps involved in the post processing phase of our code.

1. Read in file
2. Break the input string apart into new variables
3. Make any data corrections
4. Save new variables in output file
5. “Poke” or test the output file to ensure the data looks good.

Starting with reading in the file we will use textscan() because we have a date string for our timestamp.

fid = fopen(FI);  
result = textscan(fid, FMT, 'delimiter',',');  
fclose(fid);

where FI and FMT are defined as the input filename and read in format respectively. Extracting the data from the return from textscan() is pretty simple.

data\_variable = result{n};

Now we need to deal with the timestamp. The CR1000 uses a string based timestamp which is difficult to use in computational sciences, so we will convert our string timestamp to POSIX time. For brevity just accept that unix time starts at January 1, 1970, and Matlab time starts at January 1, 0000. So here’s how you make the conversion.

% Break apart the return array from textscan()  
vector\_time = result{1};  
posix\_time = datenum(vector\_time) - datenum('1970-01-01');

Saving your processed variables is pretty simple now. Matlab takes the pain out of the process by giving you the save() command. Pretty simple to use, just tell it the file name followed by the variables you want to store. We’re adding the v7.3 option just to be verbose about the file format. After the file is saved, use the whos() function to poke the file to see what the dimensions are.

% Save in output file  
save(FO, 'posix\_time', 'vector\_time', 'batt\_voltage', 'air\_tempc', ...  
 'rel\_hum', 'wind\_spd', 'wind\_max', 'wind\_dir', 'pressure', ...  
 '-v7.3')  
  
% And verify the file  
whos('-file', FO)