Learning Goals

In this lab, students will

- develop algorithms involving loops.
- carry out a top-down design process.
- modularize code, separating the algorithm from input/output statements.
- solve a problem useful in the earth sciences.

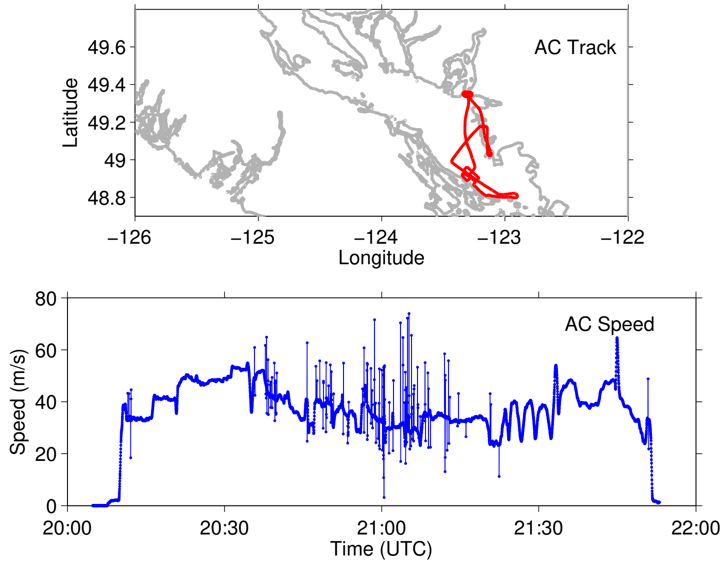

Figure 1: The aircraft dataset. The short spikes up and down, especially from 20:45 to 21:15, are clearly erroneous.

Background

In the very first lab you plotted a long sequence (a TIME SERIES) of temperatures at Sand Heads. Remember that the data showed a broad seasonal trend (colder in winter, warmer in summer), but that at very short time scales (a day or two) there was a great deal of variability which gave the curve a fuzzy look when you plotted a whole year. It is often very useful to be able to "SMOOTH" a time series to better display broad trends by averaging away the short-time variability.

A very simple way of smoothing a time series is to use a so-called RUNNING MEAN. Imagine that N data points are recorded hourly, and number them 1,2,3,. . .,$i-1, i, i+1$,. . .,N as they appear in a vector. If we are calculating a 5 point running mean, then the 5th point in the SMOOTHED time series will be the average of points 3, 4, 5, 6 and 7 from the original time series (from "two-to-the-left" to "two-to-the-right"). The 6th point in the smoothed time series will be the average of points 4, 5, 6, 7,and 8 from the original time series. The 7th point will be the average of points 5, 6, 7, 8, and 9. And so on, moving over one point each time, with your window RUNNING through the time series, calculating the average of the points in the window at each point in the original series (Fig. 2).

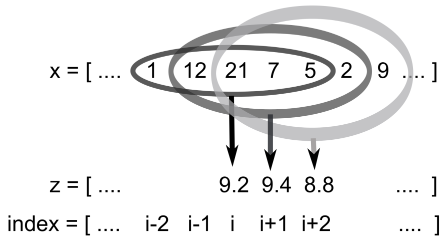
x = [.... 1 12 21 7 5 2 9 ....]
z = [.... 9.2 9.4 8.8 ....]
index = [.... i-2 i-1 i i+1 i+2 ....]

Figure 2: An example of the 5 point running mean. The ith element of z will be 9.2, an average of 1, 12, 21, 7, and 5. The next element will be 9.4, an average of 12, 21, 7, 5, and 2, and so on.

How de we describe this mathematically? Formally, we start with a time series vector $\mathbf{x}$, whose i^{th} element is x_i where $i = 1 \dots N$. Then we form a new time series $\mathbf{z}$, again with N elements, for which the i^{th} point z_i is an average of points in x within the window. If the window has a length of 5, then

$$\begin{aligned} z_i &= \frac{x_{i-2} + x_{i-1} + x_i + x_{i+1} + x_{i+2}}{5} & (1) \\ &= \frac{1}{5} \sum_{k=-2}^{2} x_{i+k} & (2) \end{aligned}$$

and if it has a length L (which has to be an odd number) then it would be

$$z_i = \frac{1}{L} \sum_{k=-W}^{W} x_{i+k}, \quad \text{where} \quad W = \frac{(L-1)}{2}. \tag{3}$$

In MATLAB you can write `x(i)` and `z(i)` as code elements corresponding to x_i and z_i, `x` for $\mathbf{x}$, etc., because MATLAB was designed to translate math into code! Now, we don't just program the long sum of 5 points in eqn. (1), because we may want to change the length of the window - we want L as another input parameter. This is because we might want a 3 point running mean, or a 9 point running mean. Instead, in this lab, you will use loops to program eqn. (3).

To write the required code, we will break the problem down into a series of steps, which you will follow as you go through the lab:

- First, develop a basic algorithm, looping through all the points in a time series of arbitrary length setting $x_i =$<something>.
- Then, develop the code that computes the running mean inside that basic loop, so that $x_i =$<running mean centered at point i >.
- Next, add some `if`-statements to handle the awkward cases that take place at the start and end of the loop.
- Finally, copy/modify that code to handle a closely related (but different) task (the running median).

Hint: it is good practice to write the code so that algorithm inputs and outputs are clearly defined using specific variable names.

The Lab

1. (a) Start writing your code in a script (i.e. an m-file) `runmean.m`. This script will itself be called by another script `runtest.m`. `runtest` should look something like this:

```
clear   % So no "junk" is left over from the last run!
% INPUTS
load lab1
x=temperature;    % input time series
winlen=25;        % the size of your window (an ODD number)

% CALCULATIONS
runmean;          % Now execute the code in runmean to do the work

% OUTPUTS
t=1:length(x);
figure(1);clf; hold on;
plot(t,x,'b');                      % original
plot(t,z,'.-r','linewi',1.5);     % Now add the output
```

   where the code in `runmean.m` will assume that `x` and `winlen` have already been defined, and will produce a `z` which is then plotted with the code at the bottom of `runtest.m`. Separating the running mean procedure from the definitions of inputs, and from output processing, is part of MODULARIZING code.

   (b) Within `runmean`, first write an OUTER loop in which the index (call it `i`) goes from 1 to `length(x)`. This outer loop is meant to process every point in the time series.

   (c) Now add an INNER LOOP (with index `k`) that implements the running mean itself. Note that MATLAB has built-in `mean()` and `sum()` functions, but DO NOT use them here. You will run into problems running your code. Why? Consider *temporarily* changing the OUTER loop limits to something that prevents these problems while debugging the code (but change it back for the next step).

   (d) Now add code to handle the END EFFECTS - those places at the beginning and the end of the dataset where we don't have `(winlen-1)/2` points to the left (or right) of the ith point - perhaps using `if` statements to perform the running mean differently. Think of two different ways of handling these effects, and implement the simplest (yes - think of two, but implement one). Describe in a comment how your TWO different solutions would work, and why you chose the one you chose.

   (e) In order to make really sure you have implemented the algorithm correctly, test it on some data for which you can work out the answer. For example, set

```
x=[1 5 3 7 9 8 4 6];
winlen=3;
```

   Work out BY HAND what the answer should be, and then see if your program replicates this.

   (f) You can also test this on a real example, like the Sand Heads air temperature data from the lab in week 2 (called `lab1.mat`). You may need to zoom in on the plot to see the two series. If this is working properly the red line will be much smoother than the blue line as daily variations are removed. Try different window lengths.

2. Now write an algorithm to determine the running MEDIAN over a window length `winlen_med` (use the built-in function `median` to replace the running mean loop). Be careful and explicitly label (with comments) what the inputs and outputs for the running median code should be. The output variable for the running median code should be named `zm`. Add this code to `runmean` so there are two outputs for the whole script: `z` and `zm`.

   Why would you use a running median? In the case of the temperature data most (or all) of the data is correct, or at least not wildly and obviously wrong. This is generally untypical of real data, which often show very intermittent "obviously erroneous" measurements for various reasons. For example, the figure on page 1 shows a speed derived from GPS positions measured on a float-plane flying a survey over the Strait of Georgia (this data can be found in the variable `gps.vel` contained in the structure `gps` which is stored in the datafile `aircraft_gps.mat`). Note that the speed tends to vary smoothly between 30 and 50 m/s, but there are strange spikes in the series (especially in the middle part of the time series when the plane is turning a lot), related to changes in the paths of radio-wave propagation when the aircraft changes its orientation and certain parts of the sky are obscured. These are clearly "bad data" because the speed of anything real won't change by 20 m/s for only one second!

   Now, if we were interested in the speed, taking a running average will not really help matters (go ahead - use your code above to see) because it will tend to pull the smooth curve away from the "correct" points. But a running median, if the window is long enough, can REJECT these OUTLIERS.

   Test your algorithm with `gps.vel` using `winlen_med=7`. When working properly the spikes will be removed from the time series.

To Hand In (due 4pm Friday Oct 12)

On CANVAS hand in your script file, called `runmean.m` exactly. I will test it by running the following code:

```
clear;
x=some_data_that_I_will_specify;            % NOT the aircraft data
winlen=some_length_not_necessarily_5;       % I'll use only odd numbersc
winlen_med=some other length;               % I'll use only odd numbers

runmean;

t=1:length(x);
figure(1); clf; hold on;
plot(t,x);       % Original time series
plot(t,z,'.-r')  % Running mean
plot(t,zm,'og'); % Running median
```

Include useful comments that describe your algorithm and also the following lines:

```
partner.name='YYYYYY';
Time_spent= XX;
```