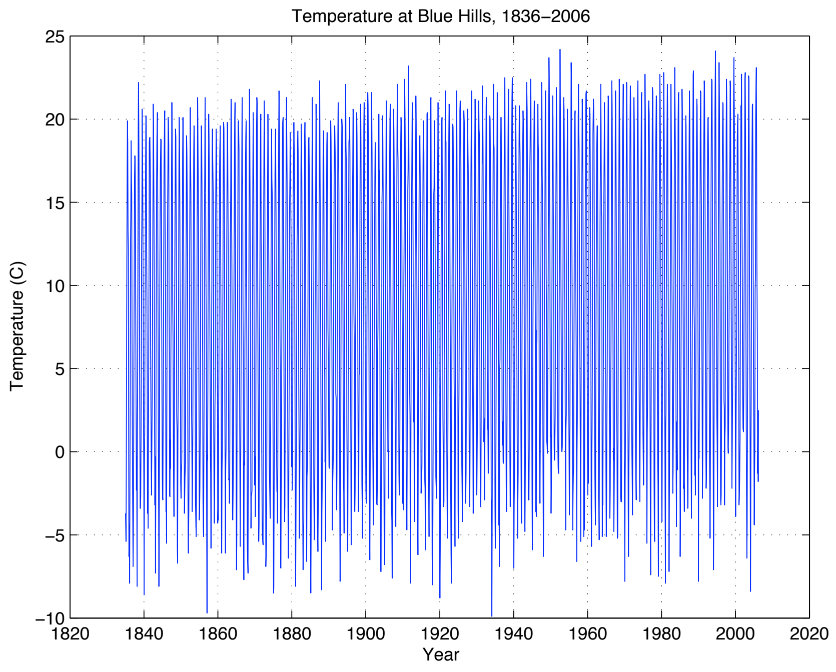
Scientific Computing

Assignment 1: Looking for Climate Change, Part 2

In the last assignment, we learned how to load up and plot some weather data from a weather observatory in the Blue Hills Reservation south of Boston. Our graph looked something like this:



If you look closely, you can see that the most prominent signal in the temperature record is the yearly seasonal cycle: it gets cold every winter and warm every summer. No surprise, but this makes it more difficult to see the long-term temperature trends. It's awfully hard to see a warming of a degree or so per century when the temperature is fluctuating by 25 degrees from winter to summer!

Removing the Seasonal Cycle

One standard technique for solving this problem is to filter out the annual cycle from the data, by plotting the "monthly temperature **anomaly**" -- that is, the difference between any given data point and the average temperature for that month over all years.

To be more precise, we want to do the following:

1) Construct an "average annual cycle", so that

annual_cycle(1) = average of temperatures in months 1, 13, 25, ... 12 n + 1 ...

annual_cycle(2) = average of temperatures in months 2, 14, 26, ... 12 n + 2 ...

and so on for all 12 months of the year.

2) Compute the “monthly temperature anomaly” for each month by taking the difference between the temperature in Month 1 and the average for that month, the temperature in Month 2 and the average for that month, and so on. Repeat for the entire data set.

3) I recommend you store your monthly averages in a 12-element vector variable (that is, a list of 12 numbers.) You will want to use a **`for`** loop to step through the temperature data, performing math on each element.

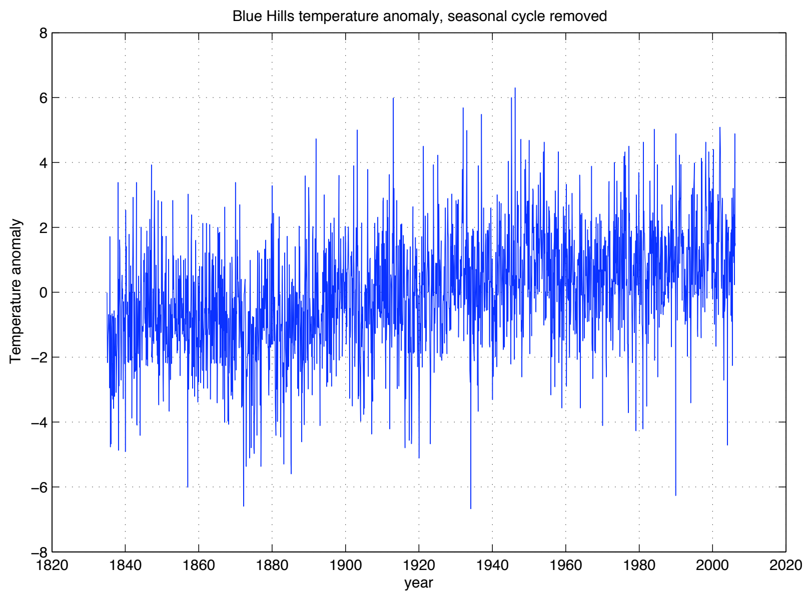
Some MATLAB commands that might be useful:

`mod(n,m)`: The modulus, or integer remainder after dividing **`n`** by **`m`**. Useful for calendar math.

`mean(y)`: Compute the mean, or average, of the vector **`y`**.

`length(y)`: The length (number of elements) of the vector **`y`**.

Question 1: Plot the temperature anomalies as a function of time. Your graph should look something like this:



The upward spikes represent unusually warm months and vice versa for cold months. You should see an upward trend in the data... however, there’s still a lot of “noise” (though it’s not measurement error, it’s real temperature variations!)

To generate a nice smooth temperature curve of the sort you might see in Al Gore’s “An Inconvenient Truth”, we can perform a “moving average”.

Moving Averages

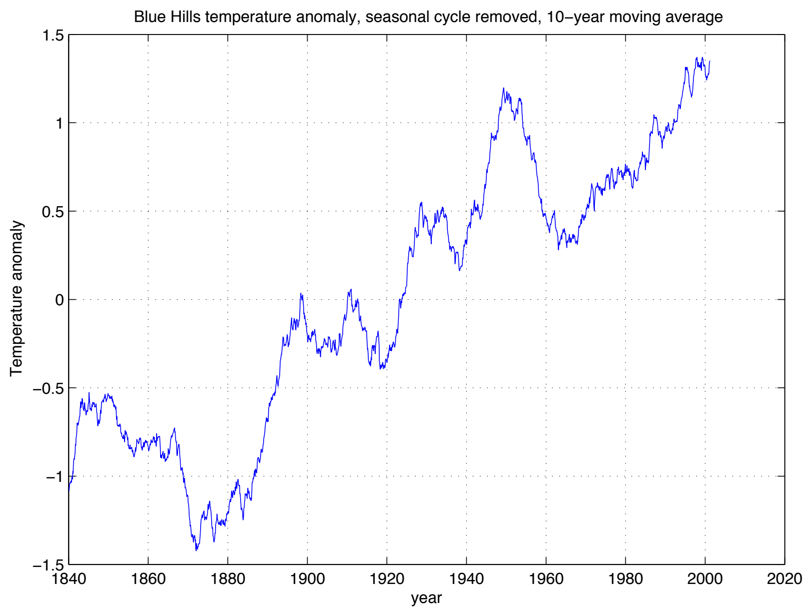
What's a moving average? If we have a series of values y(n) with n=1...N, the M-point moving average is

MA(j) = average of y(j), y(j+1), ... y(j+M)

So the 10-year moving-average temperature centered on, say, January 1910 would be the average of all 120 monthly temperature anomalies between January 1905 and January 1915. The moving average temperature centered on June 1957 would consist of the average of all months between June 1952 and June 1962, and so on.

Perform a ten-year "moving average" on the data. You'll need to write a `for` loop to make this work. You'll run into a problem at the end of the time series, when a moving average would need to use data which is not available, "off the end" of the data. Come up with a solution to that problem on your own.

Question 2: Plot the 10-year moving average of the monthly temperature anomaly data. Mine looks like this:



What happens if you use 5-year or a 1-year window for your "moving average"?

Question 3: Roughly how rapidly are temperatures in the Blue Hills increasing, in degrees C per century?

.M-file

You should write up all your code in a .m-file which does all the data processing automatically. When I simply type the name of your program, it should read the file, remove the annual cycle, perform the moving average, and plot the data.

You might wish to “partition” your code into several .m-files, each of which handle a small specific task (averaging, plotting, etc.), and a master .m-file which executes each one in turn.

Question 4: In the Scientific Computing Oncourse page, you’ll find a link “city temps”, which contains similar temperature data files for other locations. Use your .m-file to make similar plots for a few other cities. If you’ve written good general-purpose code, you shouldn’t have to change anything in your program except the file name in the `load` command.

Extra Credit: Write some code which tries to look at the *global average* temperature by loading all the temperature files and taking an average over all six locations. This is tricky because they all start and end on different dates, so you need to make sure the dates are consistent when doing your average.