Tokyo Temperature Projection Implemented in C++

Richard Boyne and Deirdrée Polak

I. Introduction

Data Source

Data was sourced from Japan Meteorological Agency[1] (JMA) which provides complete min, max and mean temperature records for every month dating back to 1876 to ± 0.1 Deg Celsius.

Numerical Methods

Given a data set, different forms of analysis were implemented:

Lagrange Interpolation - Simple method that interpolates data exactly. The extrapolation values on the other hand are extreme and unreliable due to Runge's Phenomenon[2].

Linear Regression - Using the Normal equation method with a 2x2 inversion from ACSE-7. A matrix class was made to assist with this implementation.

Standard Deviation Error - Applied to linear regression as an estimate of prediction uncertainty. The standard error of the data fit is calculated and applied to all temperatures uniformly.

Time Dependent Error - Applied to linear regression as a goodness of fit. Computes error per point with test data and fits a linear regression to predict temperature error with time.

II. Input and Outputs

The code requires input data in the form of a csv file as well as user determined parameters such as prediction method and years. These are all handled through a simple menu based user interface (see Readme.md for user manual). The user input is run through a function to ensure incorrect inputs (of say the wrong type) are processed by prompting the user for a different response.

Outputs are saved in a csv file within the local directory with time stamped names to prevent overwriting of data. Plotting is also implemented through a gnuplot pipeline; which can then be saved if desired. When the program has finished it has a restart option for user ease.

III. Code Structure

The structure of having a user interface with a menu allowed sections to be written separately (i.e. each individual choice), hence progressively building on each other (for example choice_two is called multiple times by options 3,4,5).

The code was written with the aim of being reusable; every function in the file general.cpp as well as the matrix.h class file was written with the idea of being suitable for other projects. The matrix class in particular (which is an overlay to the default vector container) allows for python like indexing of 2d arrays and overrides the multiplication operator for matrix multiplication. Other functions such as input, save_csv, load_data, get_time and plot all will clearly have a future use.

Where possible the code has also been kept with as wide a functionality as possible to allow for the user to vary there requirements. For example, even though all the data for the project has been gathered the user may still specify their own .csv file to source data from.

To improve memory performance containers were always passed by reference and vectors (subsequently matrix's) initialised to the correct lengths. However, optimisation was not a focus of the code as we only have small data sets (thus is a limitation).

IV. Code Limitations & Improvements

Conceptually our code is quite limited by the use of only simple models (linear regression or Lagrange polynomials) and as a result the assumption that global temperature is following a single simple trend dependent on one factor (time) when in fact it is very complex and dependent on several factors such as carbon dioxide emissions. Incorporating more of these factors of how they relate to each other into our model would be the next logical step.

In terms of the performance of code, due to the short time frame the code is not particularly optimised, with one of the most costly operations being reading and writing to/from csv files. Writing is particularly inefficient as data is written strait to the file rather than though an string stream buffer. Also most variables are stored dynamically when it is likely many could be replaced as static. Use of a profiler would likely reveal more potential improvements to the run time and memory usage if the code were to be further improved.

References

[1] Japan Meteorological Agency, 1-3-4 Otemachi, Chiyoda-ku, Tokyo 100-8122, Japan, https://www.data.jma.go.jp/obd/stats/data/en/smp/index.html

[2] James F. Epperson. 1987. On the Runge example. Am. Math. Monthly 94, 4 (April 1987), 329-341. DOI=http://dx.doi.org/10.2307/2323093