Homework #8 - Application of Curve Fitting to Sensor Calibration

A manufacturer has hired you to improve the accuracy of a sensor. You have been assigned to write firmware (a program) that produces more accurate results from this real device's output. Your approach is to model the sensor error function, generate a matching polynomial and then simulate the effects of your correction polynomial on the results. The hardware on the device will allow you to use up to an 8th order correction polynomial. You will need to develop analysis code (GE and Normal Mode Lease Squares using GSL) to analyze data to help produce the correction polynomial. This assignment will make extensive use of Linux pipes "|".

Problem

The example device output is not very accurate. Within the device operating range $[0.1\ \mathrm{V}, 4.9\ \mathrm{V}]$, there can be a difference of as much as 15% mid-scale error between the output of the real device and the ideal correct value. (e.g. at 2047 counts) You must get the mid-scale error as low as possible while also balancing other error metrics:

Peak percent error
Average error
Maximum count error
Average count error
Norm of residuals
R**2 error
Pearson's Correlation

ap_hw8files:

Hw8.c	A sample least squares solution framework using QR factorizations and GSL.
realDevice	Generates simulated output. The first column of data is the "ideal value", the second is the "real value". You may **ONLY** use the "real value". The first value must be passed through your correction program. You may have to set the executable bit on this binary. The output from realDevice | diffVal will become your **data.txt** needed in **hw8.c**
detError	Determines the error percentages associated with the measurements. It assumes column data of the form "ideal data" and "real data". (set executable bit if necessary)
diffVal	This takes the output of realDevice and generate an output of: "real data" and "real – ideal". Use this to generate the error function.
myplot.c	A program framework to generate a **png** figure with a plot of the device response (using gnuplot) , modify it as you see fit. Note myplot.c is NOT ANSI so don't use **–ANSI** to compile it !!
Correction.c	A stdin/stdout program framework which will contain your final correction algorithm. Your code can ONLY use the "real data" values in its calculation but must pass the "ideal value" through unchanged.

Note: If some of the programs provided do not run, change their executable bit:
chmod a+x filename (this is what make a file executable in UNIX).

Approach

1) Determine your base error statistic using the following command:

./realDevice | ./detError

In this command (read from left to right): The output of **realDevice** is piped to the input of **detError**. The output will be similar to the following:

```
./realDevice | ./detError
Max/Min/Ave/Mid Percent Error = 20.48% / 2.07% / -14.48% / 14.97%
Max/Min/Ave count error    = 420 / 16 / -265
Norm of residuals error  = 18064.12
R squared error          = 0.94430073
Pearson's Correlation    = 0.99836389
```

Note: Each student's "realDevice" will be slightly different

2) Generate your "error" data. Error data is calculated by subtracting the real data from the ideal data.

./realDevice | ./diffval > data.txt

The resulting data.txt data will have the following form:

```
ideal  error
 97   16
 98   16
 100 17
 101 17
 102 17
 103 16
 104 16
 106 17
```

3) Write "**HW8**", "GE" and "Norm Least Squares (LS)" data fitting code using the GSL libraries.

Hw8 (-ge | -norm) -o[rder] num -p[oints] file [-v[er[bose]]]

Where: -ge	-Use standard Gaussian Least Squares QR Elimination
-norm	-Use the "Normal mode" Least Squares equations
-order num	-order of the equation to use. Must be 1 or more
-point file	-data points file to evaluate
-verbose	-optional verbose flag, which dumps all major matrix and vector values.

./hw8 -ge -o 2 -p data.txt - use standard Gaussian elimination for a 2nd order equation

You need re-use your DynamicArrays code. There is HW8 section in your DynamicArrays.h code. Make any changes you need there and then use the -DHW8 compile option to enable your changes.

The output from your hw8.c code should clearly identify the solution mechanism used and the resulting polynomial, “Norm of residuals error”, “R**2 determination coefficient” and the “Pearson Correlation”.

```
e.g ./hw8 -norm  -o 2   -p data.txt
        Least Squares Solution via Norm factorization:
        f(x) =  -42.4382 + 0.231736x + -3.3201e-05x^2

        Norm of Residuals error = 2978.69
        R**2 determination coef = 0.84231330
        Pearson Correlation     = 0.91980822
```

When the –verbose mode is specified, additional detailed debug messages should be displayed, where appropriate, including: A, b, A’, A’A, A’b and a detailed list of the solution coefficients like:

```
x_ls[0] = -51.8091604955432246
x_ls[1] =  1050.7286972056774630
x_ls[2] = -32.1641519296112648
```

Sample verbose data:

```
A (3840 x 3)
  0:     1.00000     97.00000   9409.00000
  1:     1.00000     98.00000   9604.00000
 ...
3838:     1.00000   4095.00000 16769025.00000
3839:     1.00000   4095.00000 16769025.00000


b (3840 x 1)
  0:   16.00000
  1:   16.00000
 ...
3838:   84.00000
3839:   83.00000


AT (3 x 3840)
  0:     1.00000     1.00000         ...   ...
  1:    97.00000    98.00000         ...   ...
  2:  9409.00000   9604.00000   ...   ...

ATA (3 x 3)
  0:   3840.00000 8876833.00000 26378824477.00000
  1: 8876833.00000 26378824477.00000 87271905931495.00000
  2: 26378824477.00000 87271905931495.00000 305828818125810432.00000


ATB  (3 x 1)
 1018319.00000
 2838701767.00000
 8950782868241.00000
```

Note: You can verify your LS GSL code using Excel, but you must do all your real work using your LS GSL code. Excel will round coefficients by default, use “Format trend line label -> Category = scientific, decimal = 5” to see exact results

4) Create a correction polynomial "**correction**" from the results of your data fitting code. You will want to try various correction polynomials, testing each for the best final error as determined by "detError". The code must be analytic (use only mathematical equations). You must use Horner's factorization. Yay not use any sort of if-then-else logic. Your code must pass the "ideal data" through without modification. You may **not** use the "ideal data" portion for any purpose. Be sure to document all your steps in your analysis.txt file.

./realDevice | ./correction | ./detError.

Note: You will need to round floating point numbers to integers.

For positive integers:	**(int) (x + 0.5)**
For mixed data:	**x >= 0 ? (int)(x+0.5) : (int)(x-0.5)**

5) In addition to writing the correction code, you must write a gnuplot program called "**myplot**". Myplot will be used to display the ideal, before correction and after correction results. Myplot has the following syntax:

myPlot -i[n] data -o[ut] png.

"data" is the name of a four column input data file

"png" is the name of an output PNG file. .

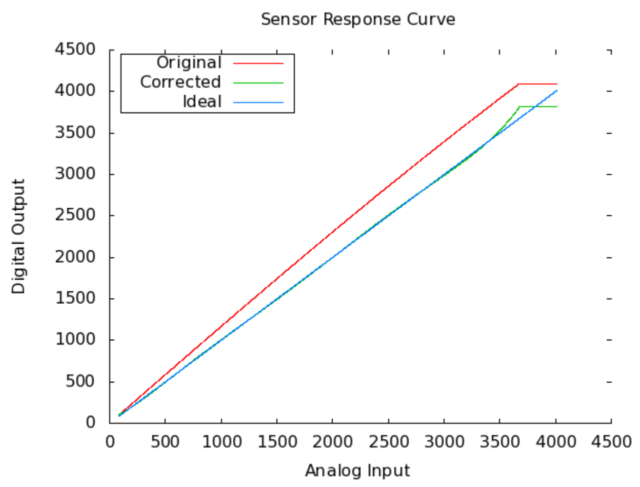
The four column data will be of the form:

idealData	realData	idealData	newData
81	97	81	101
82	98	82	101
83	100	83	103

The data file can be created from the **realDevice** output and **correction** output by using the Linux past command:

e.g: **paste realdevice.txt correction.txt > alldata.txt**

The resulting plot should have the form:



Note the axis labels, title and plot key location.
You may need to add the following to your .bashrc file to enable plots:

```
export GDFONTPATH=/usr/share/fonts/dejavu
export GNUPLOT_DEFAULT_DDFONT="DejaVuSans.ttf"
```

Makefile

You must provide a quality Makefile with the following targets: all, base, correct, plot, mem, clean and help.

"all"	-should make **hw8, correction** and **myplot.**
"base"	- should use pipes to calculate the error from **realDevice** using **detError.**
"correct"	- should use pipes to calculate the error from **correction** using **detError.**
"plot"	**-** 1) redirect the output of **realDevice** into **realdevice.txt** 2) redirect **correction** into **correction.txt** 3) paste the combined files into **alldata.txt** and then create **alldata.png** from the data.
"mem"	-should run the standard memory checking code with: ./hw8 –ge 3 data.txt (data.txt is your data you used to determine your correction equation)
help, clean	- should do the normal things

Analysis

The main effort in this assignment is *analyzing the measured data* (*i.e.*, the output of **realDevice**) to determine *what "correction polynomial" needs to be applied*. In addition to these measured values, this determination requires analysis of the "ideal" quantities versus the correct "real" values. Make sure to **document all the steps** take during this process in the **analysis.txt** file (be concise and direct to the point.)

Deliverables

All the files you create to accomplish this task should be packed in a single tar file **lastName_hw8.tar** (lastName is your last name). In addition to your C source code, you should ***include all files** you created and used to determine the applied corrections* (e.g., Matlab files, Octave files, Excel files, data.txt etc.). Also, you should produce a text file (**analysis.txt**) that describes clearly and concisely the approach taken to solve the problem and the analysis of your results

Note: Be sure to provide **YOUR data.txt** file

Grading Criteria

1. (60 pts) Correct Program
   a. (30 pts) hw8 functionality
   b. (5 pts) hw 8 memory leaks
   c. (5 pts) Correction program products acceptable maximum error reported by **detError**
   d. (5 pts) Processing pipeline implementation
   e. (5 pts) Makefile to test program
   f. (5 pts) **png** figure with curves before and after calibration.
   g. (5 pts) Correct implementation of **myplot**

2. (40 pts) Analysis
   a. (20 pts) Determination of correction equation (or 'best' solution).
   b. (20 pts) Explanation of analysis process and results.