CMPE-380: Applied Programming

Laboratory Exercise 11

Curve Fitting:

Least Squares Approximation

Table of Contents

Curve Fitting: Least Squares Approximation

Pre-Lab – 10 pts

In this lab session, we will review Least Squares data fitting and GSL implementation. Data fitting allows us to model systems with a simple polynomial and effectively "smooth" the measured data. With a fitted polynomial we can then predict the performance of the system outside the measured data points.

Review and understand the Least Squares Data Fitting class notes. In addition, using the provided **lab_a.c** framework, implement the following simple **Ax=b** GSL solution. Note, this is NOT a least squares problem but the simpler problem of solving a closed list of equations. You will be using **Ax=b** matrix composition to solve the problem.

You are provided with **lab_a.c** a frame work to implement the following polynomial.

```
//2x1 + 8x2 + 6x3 = 20
//4x1 + 2x2 - 2x3 = -2
//3x1 -  x2 +  x3 = 11
```

Compile the code with: **gcc -g -std=c99 lab_a.c -lgsl -lgslcblas -o lab_a**

1) Allocate the GSL matrix and vector using the provided defines.
2) Use gsl_matrix_set() and gsl_vector_set() to initialize the A matrix and b vectors.
3) The provided matrix and vector print routines can be used to verify your implementation.
4) Use gsl_linalg_QR_decomp() and gsl_linalg_QR_solve() to generate the solution "x".
5) Be sure to properly free all the GSL data you allocated.
6) Verify your solution matches the following:

```
Matrix A (3 x 3)
 0:     2.00000     8.00000     6.00000
```

```
1:    4.00000    2.00000   -2.00000
2:    3.00000   -1.00000    1.00000
Vector b (3 x 1)
  20.00000
  -2.00000
  11.00000

Solution vector x (3 x 1)
   2.00000
  -1.00000
   4.00000
```

Your TA will verify your **lab_a.c** file and conduct a brief oral quiz on the basic data fitting concepts. The TA may ask questions about how data fitting differs from interpolation. What the basic data fitting matrix equation is and why a simple matrix solution can't be used. The TA may ask questions about how to format the X data to implement a fitting polynomial of order N. You should be able to explain when malloc/free is used and when gsl_matrix_alloc/gsl_matrix_free should be used.

Interactive Examples – 30 pts

In this section we will reuse our lab3 Darray code as intermediate storage and then use that data to build the GSL A matrix and b vector necessary to calculate the least squares data fitting equation: AtAx = Atb. We will not build the final solver, that is your homework assignment. If you don't have Darray code, see your lab professor for a Darray object file. Please create a directory named "lab11" and change to that directory if you don't already have one.

1) Copy your **DynamicArrays.c, DynamicArrays.h** and **ClassErrors.h** file from lab3. You may have to comment out the SearchDArray() function.
2) Examine and understand the provided code **lab_b.c**, in particular review the **prepareDataForLS()** function
3) Search for "**update**" in **prepareDataForLS()** and add the necessary parameters.
4) Compile the code with:

   gcc -g lab_b.c DynamicArrays.c -DHW8 -std=c99 -lgsl -lgslcblas -o lab_b

5) Run your code, it should produce the following:

```
Reading data from testdata.txt...
The raw data in the Darray
[1.000000,2.000000]
[3.000000,4.000000]
[5.000000,6.000000]

Defining a first order solution matrix
Matrix A (3 x 2)
 0:     1.00000     1.00000
 1:     1.00000     3.00000
 2:     1.00000     5.00000
Vector b (3 x 1)
   2.00000
   4.00000
   6.00000

******* --- *******
```

```
Defining a second order solution matrix
Matrix A (3 x 3)
 0:    1.00000    1.00000    1.00000
 1:    1.00000    3.00000    9.00000
 2:    1.00000    5.00000   25.00000
Vector b (3 x 1)
   2.00000
   4.00000
   6.00000
```

6) Edit the last three lines in main() and add a forth order polynomial solution matrix feature.

7) Recompile and verify your code produces the following incremental output:

```
Defining a fourth order solution matrix
Matrix A (3 x 5)
 0:    1.00000    1.00000    1.00000    1.00000    1.00000
 1:    1.00000    3.00000    9.00000   27.00000   81.00000
 2:    1.00000    5.00000   25.00000  125.00000  625.00000
Vector b (3 x 1)
   2.00000
   4.00000
   6.00000

******* --- *******
```

8) Redirect the output to exercise.txt and show the results to your TA.

Assignment: Curve Fitting to Sensor Calibration 60 pts

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 8^{th} order correction polynomial. You will need to develop analysis code (Normal Mode Lease Squares using GSL) to analyze data to help produce the correction polynomial. This assignment will make extensive use of Linux pipes "|".
You are going to model the error and then subtract it from the input data to produce more accurate results.

Problem

The example device output is not very accurate. Within the device operating range $[0.1\ V, 4.9\ V]$, there can be a difference of as much as 28% mid-scale error between the output of the real device and the ideal correct value. (e.g. at 2047 counts) You must minimize the mid-scale while also balancing other error metrics like "Percent error".

Assignment Files

Hw11.c	A sample least squares solution framework using QR factorizations and GSL. You have to write the get_opt_long code and the specific norm lease squares code.
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 **Hw11.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	A gnuplot program that will generate PNG plots from data of the form: idealData realData idealData improvedData
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%
```

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:

```
real    error (real-ideal)
  81  124
  82  126
  83  127
  84  129
  85  131
  87  132
```

3) Write "**HW11**" "Norm Least Squares (LS)" data fitting code using the GSL libraries.

Hw11 -o[rder] num -p[oints] file [-v[er[bose]]]

Where: -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.

./Hw11 -o 2 -p data.txt - use Normal least squares gaussian elimination for a 2nd order equation

You will want to re-use your DynamicArrays code. The professor can provide a runtime DynamicArrays.o file if your code has issues. The output from your Hw11.c code should clearly identify the resulting fitting polynomial.

e.g ./Hw11 -o 2 -p data.txt

f(x) = -33.1001 + 1.66528x + -0.00017055x^2

When the –verbose flag 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] = -33.1001134254091269
x_ls[1] =   1.6652823854111758
x_ls[2] =  -0.0001705497893629
```

Sample verbose data:

```
A (3840 x 3)
 0:     1.00000    81.00000   6561.00000
 1:     1.00000    81.00000   6724.00000
 ...
3838:     1.00000   4011.00000 16088121.00000
3839:     1.00000   4012.00000 16096144.00000

b (3840 x 1)
 0:   124.00000
 1:   126.00000
 ...  (show all the rest of the data)
3838:  3944.00000
3839:  3944.00000

AT (3 x 3840)
 0:     1.00000     1.00000        ...   … (show all the rest of the data)
 1:    97.00000    98.00000        ...  …(show all the rest of the data)
 2:  9409.00000  9604.00000    ...  …(show all the rest of the data)

ATA (3 x 3)
 0:  3840.00000 7858514.00000 21027733424.00000
 1: 7858514.00000 21027733424.00000 63274317832130.00000
 2: 21027733424.00000 63274317832130.00000 203089602681098240.00000

ATB  (3 x 1)
 9373265.00000
 23965574793.00000
 70036697614275.00000
```

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. Do 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 mixed data: **x >= 0 ? (int)(x+0.5) : (int)(x-0.5)**

5) A gnuplot program called "**myplot"** has been provided . Myplot should be used to display the ideal data, before correction data and after correction data. 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.

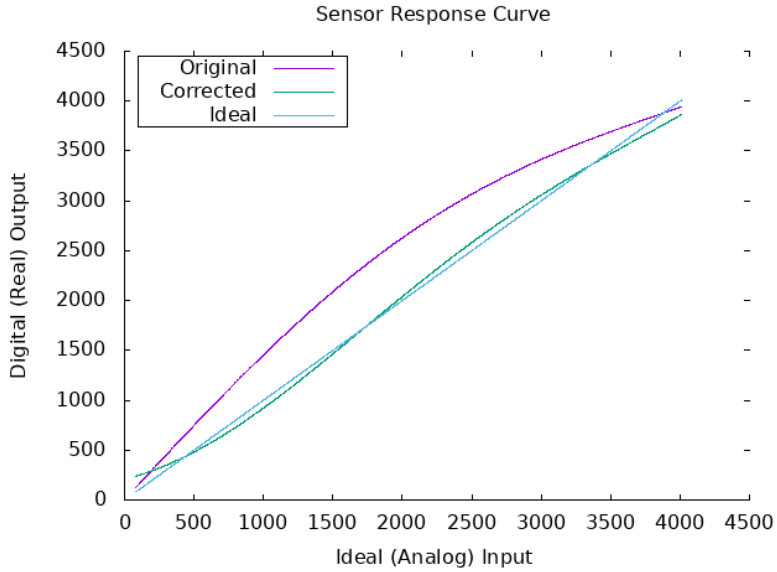
The four column data will be of the form:

idealData	realData	idealData	improvedData
81	124	81	34
82	126	82	35
83	127	83	36

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

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

The resulting plot should have the form: (yours will look slightly different)



Each graph line must be a unique color. Exact color is not critical.

Note: You may need to add the following to your .bashrc file to enable plot fonts:

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

then execute the file: **source .bashrc**

Makefile

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

"all"	-should make **Hw11** and **correction.**
"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 **realDevice | correction** into **correction.txt** 3) paste the combined files into **alldata.txt** and then create **correction.png** .
"mem"	-should run the standard memory checking code with: ./Hw11 –order 3 -p data.txt (data.txt is your data you used to determine your correction equation)
"x"	-should run an chmod +x for: realDevice, detError, myplot and diffVal
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** taken 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_Hw11.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

Grading Criteria

1. (35 points) Correct program(s)/algorithms.
2. (2 points) Memory leaks
3. (10 points) Makefile
4. (13 points) Analysis

Student Name: _____

Laboratory Grading Sheet

Lab 11 - Curve Fitting - Least Squares Approximation

Component	Point Value	Points Earned	Comments and Signatures
Pre-Lab: code	8		
Pre-Lab: quiz	2		
Interactive Exercises: GSL Example	30		
Total	40		

You must turn this signed sheet in at the end of lab to receive credit!