EEE102C++ Programming and Software Engineering II

Assessment5

Assessment Number	5
Contribution to Overall Marks	15%
Submission Deadline	23:55, May 10

How the work should be submitted?

SOFT COPY ONLY !

(MUST be submitted through ICE so that the TAs can run your programs during marking.)
Make sure your name and ID are printed on the cover page of your report.

Assessment Overview

This assessment aims at testing some basic concepts of C++ programming and initiates the routine of code development using the software development process (**SDP**), namely the five main steps of the software development process:

1. Problem statement: formulate the problem.
2. Analysis: determine the inputs, outputs, variables, etc
3. Design: define the list of steps (the algorithm) needed to solve the problem.
4. Implementation: the C code has to be submitted as a separate file. Just indicate here the name of the file.
5. Testing: explain how you have tested and verified your C++ program.

You will need to apply this methodology to each one of the following simple exercises.

What should be submitted?

A short ***report*** (up to a few pages of texts plus C++ source codes) detailing for all the questions of the assignment. The answer for each question should follow the SDP method:

a) SDP steps 1 to 3. (30% of the total marks for that question)
b) SDP step 4 (implementation): your C++ source code including the comments. (50%)
c) SDP step 5 (testing): you will explain how you have tested the correctness of your C++ program and will include some sample runs of your C++ Programs. (20%). **Testing result must be shown by screenshot.**

The report in Microsoft Word format **(.DOCX file)**and **C++ source code (with comments)**, for all questions should also be zipped into ***a single file***. (It is a good practice to include comments in your code stating the aim of the program, what are the inputs, what are the outputs, which algorithm is used, who is the author and so on.)

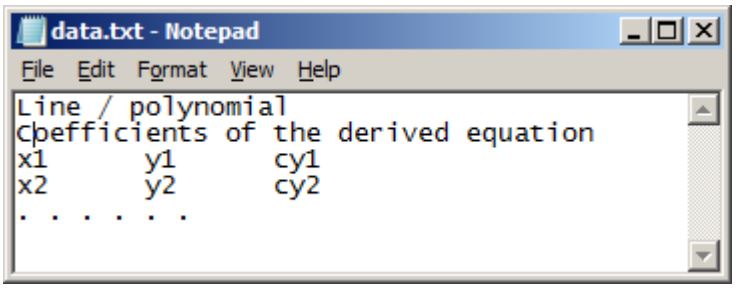
EXERCISE 1 (15 POINTS OUT OF 15)

In this part, a programme is required to fit curves, using a regression analysis (as shown in Appendix), to data sets read from a text file. A set of x,y data pairs are to be read from a file (each line of the file containing an x value and a y value separated by a space or TAB).

- The user should be able to select between fitting a straight line or a second order polynomial equation to the data pairs.
- The coefficients of the best fit curve to these points are then calculated. The programme should produce an output file and show a message to the user when it finishes a fitting process.

The output file should have the following layout

1. The first line of the file indicates the type of equation derived (line or polynomial);
2. The second line of the file contains the coefficients of the best fit curve.
3. From the third line, each line contains an original x value, an original y value and a calculated value of y from the best fit equation for the original x). The values need to be separated by a TAB. The format of the output file :



After you obtain the output data file, **open it with Microsoft Excel to plot the original discrete data and those from the fitting as a curve.You do not need to use your programme to plot the curves!**

- The user can use both the straight line and the second order polynomial equations. For each original x, there are two calculated values of y (CLy and CSy). CLy is calculated by the straight line equation, CSy is calculated by the second order polynomial equation. Accumulated absolute values of errors between CLy and y ($E_L = \sum_x |CLy - y|$), CSy and y ($E_S = \sum_x |CSy - y|$) can be calculated. To compare E_L and E_S, the smaller one will be indicated as the better fitting.

Appendix: Curve fitting algorithms

Given a set of x,y data pairs it is often necessary to automatically calculate an equation which gives the best fit line through the data. This type of analysis is known as curve fitting or regression analysis.

The mathematical basis behind curve fitting is quite straightforward. You have a set of data pairs represented by (x1,y1), (x2,y2) ..... (xn, yn). You want an equation which represents this data, the exact type of equation will depend on the way in which the data varies (there is no point trying to fit a straight line to a set of data which is obviously not straight). Given any mathematical relationship between x and y it should be possible to generate a curve fitting algorithm. Two examples are discussed here.

Straight line

The equation for a straight line is: y = ax + b. Suppose that we have n pairs of (x,y) values from the text file, the values of a and b for the best fit line to the data pairs can be obtained from the following equation

$$a = \frac{\begin{vmatrix} b_1 & d_1 \\ b_2 & d_2 \end{vmatrix}}{\begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix}} \qquad , b = -\frac{\begin{vmatrix} a_1 & d_1 \\ a_2 & d_2 \end{vmatrix}}{\begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix}}$$

where

$$a_1 = \sum_{i=1}^{n} (x_i * x_i), \quad a_2 = \sum_{i=1}^{n} x_i ,$$

$$b_1 = \sum_{i=1}^{n} x_i , \qquad \mathrm{b}_2 = \mathrm{n} ,$$

$$d_1 = -\sum_{i=1}^{n} (x_i * y_i), \qquad d_2 = -\sum_{i=1}^{n} y_i$$

The value of a second order determinant $\begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix}$ is the difference between the two cross products (a_1b_2 - a_2b_1). It is easy to create a C++ function to calculate its value.

Second order polynomial

If we wish to fit the curve: $y = ax^2 + bx + c$ to the data pairs, we can follow a similar approach to that for a straight line. The values of a, b and c for the best fit curve can be calculated from the following equation

$$a = -\frac{\begin{vmatrix} b_1 & c_1 & d_1 \\ b_2 & c_2 & d_2 \\ b_3 & c_3 & d_3 \end{vmatrix}}{\begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix}} \quad , \quad b = \frac{\begin{vmatrix} a_1 & c_1 & d_1 \\ a_2 & c_2 & d_2 \\ a_3 & c_3 & d_3 \end{vmatrix}}{\begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix}} \quad , \quad c = -\frac{\begin{vmatrix} a_1 & b_1 & d_1 \\ a_2 & b_2 & d_2 \\ a_3 & b_3 & d_3 \end{vmatrix}}{\begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix}}$$

where

$$a_1 = \sum_{i=1}^{n}(x_i * x_i * x_i * x_i), \quad a_2 = \sum_{i=1}^{n}(x_i * x_i * x_i), \quad a_3 = \sum_{i=1}^{n}(x_i * x_i)$$

$$b_1 = \sum_{i=1}^{n}(x_i * x_i * x_i), \qquad b_2 = \sum_{i=1}^{n}(x_i * x_i), \quad b_3 = \sum_{i=1}^{n} x_i$$

$$c_1 = \sum_{i=1}^{n}(x_i * x_i), \qquad c_2 = \sum_{i=1}^{n} x_i, \qquad c_3 = \mathrm{n}$$

$$d_1 = -\sum_{i=1}^{n}(x_i * x_i * y_i), \qquad d_2 = -\sum_{i=1}^{n}(x_i * y_i), \qquad d_3 = -\sum_{i=1}^{n} y_i$$

Solving a third order determinant is fairly simple in that the solution can be broken down to utilise a function written to solve second order determinants:

$$\begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix} = a_1 \begin{vmatrix} b_2 & c_2 \\ b_3 & c_3 \end{vmatrix} - b_1 \begin{vmatrix} a_2 & c_2 \\ a_3 & c_3 \end{vmatrix} + c_1 \begin{vmatrix} a_2 & b_2 \\ a_3 & b_3 \end{vmatrix}$$

You can use a C++ function to calculate the value of a third order determinant. The 9 elements in the determinant can be passed into the function by a two-dimensional array.