2G03 Homework 8, 2020

Due: Tuesday, Dec 1st

The files are available on phys-ugrad in `/home/2G03/HW8`

You should submit C++ code and write Makefiles for the exercises marked with `PROGRAM`. **Please follow the guidelines described on the course website**. You should create a directory called *HW8* in your area for these programs. The grader will look there to locate and run your programs.

Arrays – File IO – Interpolation

Exercise 1 `PROGRAM` *Numerical Interpolation*

Imagine you have a table of data (x_i, y_i) but you need a smooth curve $y(x)$ such as to integrate an ODE. The data might be growth rates, chemical reaction rates, profits or disease death risk etc...

Thus you need to get function $y(x)$ for any x based on a list of y_i values for specific values of x_i. If you are lucky, the data have regularly spaced x_i values and corresponding y_i values. It is then easy to use an *interpolation* scheme to estimate values for y for intermediate values of x that are between values listed in the table.

In this exercise, we will develop a program that reads in a table of data points, (x_i, y_i), and interpolates to estimate y for values of x in between those actually in the table.

1.1 Given two points with co-ordinates ($x1$,$y1$) and ($x2$,$y2$), there is a unique straight line connecting them of the form $y = mx + c$, where m is the gradient and c is a constant. Write down expressions for the coefficients, m and c, in terms of the co-ordinates of the points.

1.2 Write a float function called `interpolate` that takes two pairs of float coordinates and an intermediate x value (a total of 5 float arguments: x1, y1, x2, y2 and x). The function should return a y value based on a straight line fit of the points. Incorporate your function into a file `interp.cpp`.

Consider if your function needs to check the input values in any way and add any checking you consider necessary.

1.3 Write a short test program in a file called `testinterp.cpp` to test your function. Have your program print out what your `interpolate` function returns for the points, ($x1$,$y1$)=(3, 13.75), ($x2$,$y2$)=(4, 19) and an x value of 3.25.

1.4 Write a new float function called `lookup` for your `interp.cpp` file that takes two float arrays and a float number as arguments. The first array will contain x_i and the second will contain the corresponding y_i values. The float number given is the x value for which an interpolated result is desired. You will need to write code to locate the index, i, such that $x_i < x < x_{i+1}$. Keep your code general, without making any assumptions about the values, x_i, other than that they are increasing order.

Remember to do something appropriate if x has a bad value. You may find that just 3 arguments to the function `lookup` is not enough to do this. You probably want to test your

function with some prints at this stage just to make sure the index chosen is correct (but don't leave those prints in the final version).

1.5 If you can assume that the x_i values are uniformly spaced and you know the spacing, is there a fast and simple way to find the right index? Note: Don't change your code.

1.6 To complete your `lookup` function, once you have located the right index, you can use the two points on either side of x to interpolate and get a y estimate. Call your `interpolate` function to return an estimate of the y values for the x value given.

Add some code to your test program so that it uses the following arrays and calls your `lookup` function with the values, $x = 3.25$, 4.31 and 0.55. Save and hand in the output from your program. Note that the printout you hand in should be indentical to what the T.A. sees when running your program. Note that the output $x = 3.25$ should be the same as in the previous test.

```
float x[] = { 0,    1,  2,     3,  4 };
float y[] = { 7, 7.75, 10, 13.75, 19 };
```

1.7 Write a new main program in a file `lookuptable.cpp` that reads in the array data from a data table provided in a file such as `/home/2G03/HW8/input.dat`.

Note the format of the file. The first line has an integer indicating the number of data points. Each line after that has one pair of values, (x_i, y_i) representing each data point. After you have read the arrays in, print out some values to make sure you read them correctly.

As you may have guessed, the data in `input.dat` was generated from the same function as the test points already given. Have your program read in test x values and respond appropriately with an estimate for y at that x value from your `lookup` function that is based on the arrays read in from `/home/2G03/HW8/input.dat`.

Run your program and enter in the values, $x = 3.25$, 4.31, 0.55, 10.04 and 9.99. Save and hand in the output from your program for these values.