CS 109 – C/C ++ Programming for Engineers w. MatLab– Summer 2012
Homework Assignment 4
Functions Involving Barycentric Coordinates and Files

Due: **Wednesday July 11th by 8:00 a.m., via Blackboard.** Optional hard copy may be turned in during lab on Thursday.

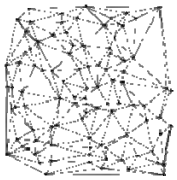
Overall Assignment

For this assignment you are to write several functions for interpolating wind speed and temperatures based on collected data read in from files, using barycentric coordinates, and a main( ) program to drive the functions.

Background: Triangulation (from HW1)

Engineers and scientists are often faced with the problem of interpolating function values over an area, based on measurements taken at irregularly spaced locations. For example, weather data is measured principally at airports, which are distributed somewhat randomly about the country, and oil well test sites are also irregularly spaced. Modern computer graphics is also based on interpolating color values between the vertices of triangle meshes.

One approach to this problem is to first ***triangulate*** the space based on the known data points, which basically means to subdivide the space into triangles as shown in the following diagram:



Delaunay triangulation of a random set of 100 points in a plane. (Source: http://en.wikipedia.org/wiki/Delaunay_triangulation)

There are many different algorithms for triangulating a space, all of which are beyond the scope of this assignment.

Background: Barycentric Coordinates (from HW1)

Once a triangulation has been determined, and the particular triangle that contains the point of interest has been identified, then the next step is to interpolate the function values from the vertices of the triangle to the point of interest. This is where barycentric coordinates come in.

Given a triangle defined by vertices r_1, r_2, and r_3, then any point r interior to the triangle can be defined by a weighted sum of the three vertices, as given by the equations:

$$r = \lambda_1 r_1 + \lambda_2 r_2 + \lambda_3 r_3$$

where λ_1, λ_2, and λ_3 are the barycentric coordinates, subject to the constraints:

$$\lambda_1 + \lambda_2 + \lambda_3 = 1.0$$

$$0.0 \leq \lambda_i \leq 1.0 \ \forall i \text{ in } 1, 2, 3 \text{ for points interior to the triangle.}$$

A function f whose values are known at the vertices can be interpolated as:

$$f(r) = \lambda_1 f(r_1) + \lambda_2 f(r_2) + \lambda_3 f(r_3) \qquad (1)$$

Calculation of Barycentric Coordinates (from HW2)

Given the Cartesian coordinates of a point of interest, (x, y), and the Cartesian coordinates of the three vertices of a triangle, ((x_1, y_1), (x_2, y_2), and (x_3, y_3)), the barycentric coordinates of the point can be calculated as[1]:

$$\lambda_1 = \frac{(y_2-y_3)(x-x_3)+(x_3-x_2)(y-y_3)}{(y_2-y_3)(x_1-x_3)+(x_3-x_2)(y_1-y_3)} \quad (2)$$

$$\lambda_2 = \frac{(y_3-y_1)(x-x_3)+(x_1-x_3)(y-y_3)}{(y_2-y_3)(x_1-x_3)+(x_3-x_2)(y_1-y_3)} \quad (3)$$

$$\lambda_3 = 1 - \lambda_1 - \lambda_2 \quad (4)$$

Classification of Points Based on Barycentric Coordinates (from HW2)

Given the Barycentric coordinates of a point of interest the point may be classified as follows:

Conditions	Classification
$0.0 < \lambda_i < 1.0 \; \forall$ i in $1, 2, 3$	Interior. (Note strict less-than operators)
One $\lambda = 0.0, 0.0 < \lambda_i < 1.0$ for the other 2 λs	Edge.
One $\lambda = 1.0$, The other two λs $= 0.0$	Vertex
At least one $\lambda < 0.0$ *or* > 1.0	Exterior

Note: It may be more efficient to check for the classifications in an order other than the one presented in the table above.

Data File and User Input

For this assignment you will be reading data in from a single data file, having the following file formats:

Weather Data File (e.g. NOAA_Data2.txt) - This file has five columns of numbers, consisting of vertex indices in the first column, x-Coordinates (Longitudes) in the second column, y-Coordinates (Latitudes) in the third column, wind speed (mph) in the fourth column, and temperature (deg F) in the fifth column. Each set of three lines corresponds to a single triangle. (I.e. the first three lines define the first triangle in the file, the next three lines the second triangle, etc.) The vertex indices correspond to the vertex indices from HW2.

Your program will ask the user for the name of the input data file and the X and Y Cartesian coordinates of a point at which function values are desired. All other necessary values will be calculated or read in from the data file.

[1] Source: http://en.wikipedia.org/wiki/Barycentric_coordinates_(mathematics)

Functions Needed

For this assignment the following functions will need to be written:

- `double interpolate( double f1, double f2, double f3, double lambda1, double lambda2, double lambda3, int & errorCode );`

  This function basically implements equation (1) above, returning the interpolated function value. errorCode should be set to zero if there are no problems, or a negative value if there is any problem with the input. (E.g. if the sum of the lambdas is not equal to 1.0. However you should not test for exact equality – Check to see if 1.0 minus the sum of the lambdas is within a small tolerance of 0.0, where a typical tolerance may be 1.0E-6. Note that you need an absolute value function for this calculation.)

- `int calcBarycentricCoordinates( double x1, double y1, double x2, double y2, double x3, double y3, double x, double y, double & lambda1, double & lambda2, double & lambda3 );`

  This function calculates the barycentric coordinates, using equations (2), (3), and (4) above. The return code is a positive integer classifying the point as interior, exterior, edge, or vertex. In the event of an error, (e.g. if the three vertices of the triangle are co-linear, indicated by a denominator of zero in equation (2) and (3)), then all the barycentric coordinates should be set to 0.0 and the function should return a negative value. The defined constants for the return codes should be declared globally, either as const ints or as a globally defined enumerated type.

- `int getVertexDataFromFile( istream & infile, double & x1, double & y1, double & x2, double & y2, double & x3, double & y3, double & wind1, double & wind2, double & wind3, double & temp1, double & temp2, double & temp3, int & index1, int & index2, int & index3 );`

  This function attempts to read data for three vertices in from the open data file, and store the values in the appropriate variables. The return value is the number of vertices successfully read in from the file, which should be 3 unless the end-of-file is encountered or some other error occurs. This function does not need to open or close the file, or ask the user for any input.

Program Details

For this assignment the majority of the calculations are moved into functions, which means the primary task of main( ) is to handle input and output, open and close files, and call the functions in the proper order with the proper arguments and check and report the results. More specifically, main should carry out the following tasks:

- Your program should first print out your name and ACCC account name (e.g. astudent), and explain to the user what the program does.

- Your program should then ask the user to enter three pieces of information: The name of the input data file, and the X and Y coordinates of a point at which function values are

desired. You should verify that you can open the data file properly and possibly ask the user to enter a new filename if the file failed to open before asking for the X and Y coordinates.

- Now main should begin a loop, looking through all the triangles in the input data file to try and find one for which the specified coordinates are either interior, edge, or a vertex, and if found, using that triangle to interpolate and report the wind speed and temperature at the specified point. More specifically, main should:

    1. Call getVertexDataFromFile to read in the coordinates, function values, and indices corresponding to the three vertices of a triangle. If the function does not return a value of 3, then break out of the loop.

    2. Call calcBarycentricCoordinates to calculate the barycentric coordinates of the point within the triangle, and determine if it is interior, exterior, edge, vertex, or an error. If it is not interior, edge, or vertex, then skip the remainder of this loop and start the next iteration.

    3. Call interpolate (twice) to determine the wind speed and temperature at the point of interest. Report the results and break out of the loop. (Also report the vertex indices of the triangle in which the point was found, and what type of point, e.g. interior, edge, or vertex, as well as the barycentric coordinates. It is probably also a good idea to go ahead and report all the data for the three vertices of the triangle.)

- If no triangle could be found for which the point was interior, edge, or vertex, then main( ) should report that fact, and report how many triangles were examined.

- main( ) should close the data file before ending.

Special Notes:

- **You should work out some sample problems by hand before writing any computer code.** For this problem it is probably a good idea to plot out the triangle vertices on graph paper, so you have some idea as to whether the coordinates should be interior, exterior, edges, or vertices. (The map and excel spreadsheet provided may be useful in this regard.) For each result your program calculates you should look at the function values at the vertices and your reported results, and ask yourself if the result seems reasonable. (For interior points the interpolated value should lie somewhere between the minimum and maximum vertex function values. That is not necessarily true for exterior points.)

Evolutionary Development

When dealing with large complex programs, it is generally best to work one step at a time, getting part of the program working and tested before attempting the next part. For this assignment it is probably best to develop and test each of the functions individually before trying to incorporate them into a finished program. For example, you could modify a copy of your program for HW1 to call the interpolate function instead of doing the calculations directly, after asking the user for the necessary input. Then when you know the interpolate function works right, you can copy it into your HW4 program.

What to Hand In:

- Your code, **including a user documentation file,** should be handed in electronically using Blackboard.
- The intended audience for the documentation file is a general end user, who might want to use this program to perform some work. They do not get to see the inner workings of the code, and have not read the homework assignment. You can assume, however, that they are familiar with the problem domain (e.g. the data to be interpolated.)
- A secondary purpose of the documentation file is to make it as easy as possible for the grader to understand your program. If there is anything special the grader should know about your program, be sure to document it in the documentation file. In particular, if you do any of the optional enhancements, then you need to document what they are and anything special the TA needs to do to run your program and understand the results.
- If there are problems that you know your program cannot handle, it is best to document them as well, rather than have the TA wonder what is wrong with your program.
- Make sure that your name appears at the beginning of each of your files. Your program should also print this information when it runs.

Optional Enhancements:

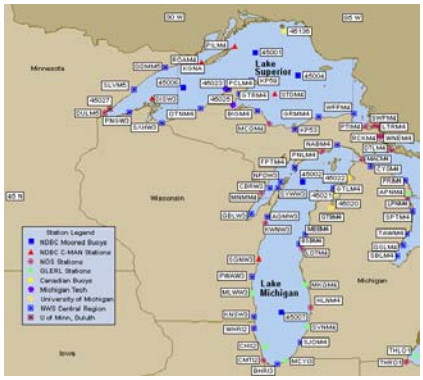
It is course policy that students may go above and beyond what is called for in the base assignment if they wish. These optional enhancements will not raise any student's score above 100 for any given assignment, but they may make up for points lost due to other reasons.

- If a given data point is external to all of the triangles in the file, try to identify which of the triangles is "closest" to the point. One possible criteria is to look for the triangle with the smallest absolute sum of deviations of lambda from the range 0.0 to 1.0. (I.e. a lambda of 1.6 has a deviation of 0.6, and a lambda of -0.8 has a deviation of 0.8, while any lambda between 0.0 and 1.0 has a deviation of zero. Find the sum of the deviations for each triangle, and then identify which triangle has the smallest such sum.) You could go ahead and estimate the wind speed and temperature based on that triangle, with the warning to the user that it is an extrapolation instead of an interpolation.
- Other enhancements that you think of – Check with TA for acceptability.

Future Work:

Once we learn about functions, we will be breaking the work in this assignment up into separate functions – One to calculate the barycentric coordinates and one to classify points at a minimum. Once we learn about files and arrays, we will start working with a network of triangles, with the triangle vertices and the list of which vertices make up each triangle read in from data files. We can then use the classification function to identify which of our triangles contains the point of interest, and then interpolate function values based on the vertex values and barycentric coordinates, as in HW1.

For future assignments we will be using the concept of barycentric coordinates and triangulation to interpolate and plot weather conditions based on data from NOAA weather buoys, as shown in the following map, most likely limited to the southern half of Lake Michigan, e.g. stations SGNW3 to MKGM4 and below.


Lake Superior
Lake Michigan
Minnesota
Wisconsin
Michigan
Station Legend
NDBC Moored Buoys
NDBC C-MAN Stations
NOS Stations
GLERL Stations
Canadian Buoys
Michigan Tech
University of Michigan
NWS Central Region
U of Minn, Duluth