CS 109 – C/C ++ Programming for Engineers w. MatLab– Summer 2012
Homework Assignment 5 – Second Draft
Barycentric Coordinates and Arrays

Due: **Wednesday July 18th by 8:00 a.m., via Blackboard.** Optional hard copy may be turned in during lab on Thursday.

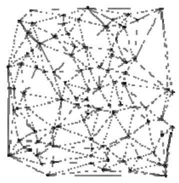
Overall Assignment

For the final C++ assignment on barycentric coordinates, we will be employing arrays for storing and passing vertex data, triangulation configurations, and barycentric coordinates. It will now be much easier to loop through a complete list of triangles to find a suitable candidate for any set of coordinates within range.

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:

$$f(r) = \lambda_1 f(r_1) + \lambda_2 f(r_2) + \lambda_3 f(r_3) \tag{1}$$

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]:

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

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

$$\lambda_3 = 1 - \lambda_1 - \lambda_2 \tag{4}$$

Classification of Points Based on Barycentric Coordinates (refined from HW2)

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

Conditions	Classification
At least one $\lambda < 0.0$	Exterior
Otherwise, any $\lambda = 1.0$	Vertex
Otherwise, any $\lambda = 0.0$	Edge.
Otherwise, by elimination, $0.0 < \lambda_i < 1.0 \ \forall$ i in $1, 2, 3$	Interior. (Note strict less-than operators)

Note: The most efficient order to check for the classifications is that given in the table above, using an if-else structure.

Data Files and User Input

For this assignment you will read data in from two data files, having the following file formats:

Weather Data File (e.g. NOAA_Data3.txt) - This file has the same format as NOAA_Data.txt presented in HW3, specifically four columns of numbers, consisting of x-Coordinates (Longitudes) in the first column, y-Coordinates (Latitudes) in the second column, wind speed (mph) in the third column, and temperature (deg F) in the fourth column.

Triangulation Data File (e.g. triangulation3.txt or triangulation4.txt) - This file contains triangulation information in the following format:

- The first line of the file contains a single integer, indicating how many triangles are listed in the file.
- The next lines contain 3 integers per line, indicating which vertices make up particular triangles. Note that these indices are zero-based, so the number 0 refers to the first vertex (in the weather data file), the number 5 refers to the 6th vertex, etc. There should be exactly as many of these lines as the value of the first integer in the file.

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

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

Functions Needed

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

- `int getVertexData ( istream & infile, double xy[ ][ 2 ], double wind[ ], double temp[ ], int size );`

  This function attempts to read vertex data 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 a positive integer less than or equal to size unless some error occurs. This function does not need to open or close the file, or ask the user for any input.

  - size indicates the maximum space available in the arrays. Your code should not read in any more than this many lines.
  - The first column of the xy array are the x-coordinates, and the second are the y-coordinates.

- `int getTriangulationData ( istream & infile, int triangles[ ][ 3 ], int size );`

  This function attempts to read triangulation data from the open data file, and store the indices in the triangles array. The return value is the number of triangles successfully read in from the file, which should be a positive integer less than or equal to size unless some error occurs. This function does not need to open or close the file, or ask the user for any input.

- `int calcBarycentricCoordinatesFromList( double xy[ ][ 2 ], int triangles[ ][ 3 ], int index, double xCoord, double yCoord, double lambda[ ] );`

  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.

  - The xy and triangles arrays are the full arrays read in from the data files.
  - "index" indicates which triangle in the triangle list to calculate the barycentric coordinates for.
  - xCoord and yCoord are the Cartesian coordinates of the point for which the barycentric coordinates are desired.
  - lambda is filled in by the function with the calculated lambda values.

- `double interpolateFromList( double f[ ],`
  `int triangles[ ][ 3 ], int index, double lambda[ ],`
  `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.)

  The arrays for f and triangles contain the full list of data read in from the data files, and index indicates which triangle corresponds to the given lambdas.

Program Details

For this assignment, 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 for the file names for the weather data and triangulation configuration. You should verify that you can open and read the data files properly and possibly ask the user to enter a new filename if a file failed to open or if the function was unable to read the data. (getVertexData and getTriangulationData should be called at this stage.)

- Next main should ask for the X and Y coordinates for which the interpolation is desired.

- Now main should begin a loop, looking through all the triangles in the list 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 calcBarycentricCoordinatesFromList (or calcBarycentricCoordinates, see below), 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 exterior, then skip the remainder of this loop and start the next iteration, looking at the next triangle on the list.

  2. Call interpolateFromList (or 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.

Alternative Function Options

For calculating the barycentric coordinates and performing the function interpolations, the "FromList" versions described above are the better way to go. That approach passes the entire arrays read in from the data files, and uses indices and indirection to select specific rows. If you can understand the indirection, then that is the approach you should take.

However the use of indirection can be confusing, so an alternative is also offered, which is to write the following two functions instead:

- `int calcBarycentricCoordinates( double xy[ ][ 2 ], double xCoord, double yCoord, double lambda[ ] );`
- `double interpolate( double f[ ], double lambda[ ], int & errorCode );`

  These functions act the same as their similarly-named predecessors, except that the xy and f arrays only contain three entries, corresponding to the three vertices of a single triangle, and there is no triangle list passed to the functions. This makes these functions easier to write, but requires more work on the part of main( ), in order to build the smaller arrays passed to these functions.

Another alternative would be to write both the FromList and non FromList functions, where the FromList versions are "wrappers" or "adapters", that only build the small arrays from the big arrays, and then call the non FromList versions listed here to do the actual work.

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.

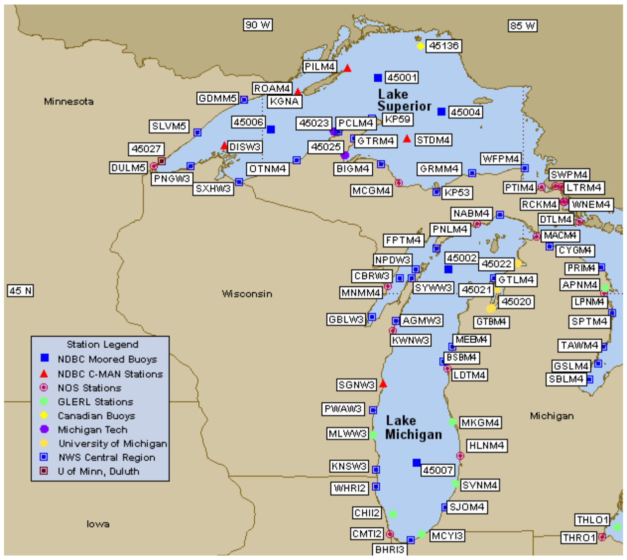
Optional Enhancements:

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.

- Allow the user to solve multiple problems by specifying additional coordinates. This is a fairly minor enhancement at this point.
- 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 criterion is to look for the triangle with the largest minimum lambda. (Any triangle for which the point is exterior will have at least one and possibly two negative lambda values. Look for the triangle whose most negative lambda is closest to zero. Alternatively you could look for the triangle for which the sum of all negative lambdas is closest to zero.) 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. Note that the work of finding the closest triangle should probably be done in a new function.
- Other enhancements that you think of – Check with TA for acceptability.

Data Source:

The data for this assignment comes from the NOAA weather buoy system, downloaded from http://www.ndbc.noaa.gov/maps/WestGL.shtml


90 W
85 W
45 N
Minnesota
Wisconsin
Michigan
Iowa
Lake Superior
Lake Michigan
45136
PILM4
45001
ROAM4
KGNA
GDMM5
45004
45006
45023
PCLM4
KP59
SLVM5
GTRM4
STDM4
45027
DISW3
45025
WFPM4
DULM5
OTNM4
BIGM4
GRMM4
SWPM4
PNGW3
SXHW3
MCGM4
KP53
PTIM4
LTRM4
RCKM4
WNEM4
NABM4
DTLM4
PNLM4
MACM4
FPTM4
CYGM4
NPDW3
45002
45022
PRIM4
CBRW3
GTLM4
APNM4
MNMM4
SYWW3
45021
45020
LPNM4
GBLW3
AGMW3
GTBM4
SPTM4
KWNW3
MEBM4
TAWM4
BSBM4
GSLM4
LDTM4
SBLM4
SGNW3
PWAW3
MKGM4
MLWW3
HLNM4
KNSW3
45007
WHRI2
SVNM4
SJOM4
CHII2
THLO1
CMTI2
MCYI3
THRO1
BHRI3
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