Comments on HW 4.

This is the first HW where the result was not easily determinable (e.g. a correct figure). Here is the procedure for HW where you have to write a program like this one. You need to turn in your best attempt by the due date, but all programming homeworks are correctable to 100% - this means that after I look at your code you can fix it till it works or meets the specified conditions for all homeworks (stuff in bold below). So you can get a perfect score for the homework part of the course. Don’t use this procedure to let things slide however as it will just get harder and harder to keep up.

So we will have a programming homework for next week, **due Tue., Oct** **8**. This is quick turnaround, but it was a mistake to extend the previous homework by a week as it just pushed the cramming to get it done off. I will look at what you have and get back to you with changes till each homework is complete.

Learning to program and learning new programming languages is just like learning another human language – you need to practice and practice and practice. This course is your opportunity to do this.

**Put your name in each file you hand in.**

**Document the input/output** (what things are – e.g. latitude and longitude in degrees; whether it is a scalar, vector, matrix; etc.)

The back-azimuth not simply related to azimuth (in general it is not the az±180). Consider a string making a great circle between two points on a sphere (under the condition that they are not antipodal) – pick one end as being the location of the earthquake. The azimuth from that point to all points along the great circle is a constant. As you move along the string on that great circle, however, the back azimuths from the points on that great circle are not constant. To calculate the back azimuths, you can just change order of the input vectors and call your function again, or you can calculate the back azimuth by multiplying the cross product by -1, since **X**x**Y**=-**Y**x**X**, and doing the calculation for  with respect to the vector at the other end (this part is the key).

There is no need to pass the routine the number of stations or earthquakes or pairs of points. You can get this information from the matrix sizes. As long as they match up (one station and one earthquake; N stations and one earthquake, or M “station” points and M “earthquake” points it is easily vectorizable. You can do M “station” points and N “earthquake” points but you have to loop and it is easier to do the loop from outside the function).

**Use the degree form of trig functions** – sind, cosd, tand, asind, acosd, atan2d (to get in correct quadrant). Having to convert from degrees to radians and back is a lot of extra coding (this is not a typing class) and chances to make errors. MATLAB initially only took radians for the trig functions, but it has been many years since they introduced the degree version.

You need to use both the sin and cos to get the 4-quadrant azimuth. The hardest way is to use the signs of the sin and cos to find the quadrant, the next hardest way is to use the result from 2 quadrant atand and the sign of the sin to set the quadrant, or use atan2d that gives you the answer directly in the correct quadrant.

**Vectorize** (most did) the function if it makes sense to do so (in this case it did), but it is not well documented (the documentation needed is in the description of the input and output parameters).

**Test your results.** Use the MATLAB functions *distance* and *azimuth* to check your answers.

Make sure it is a function (what was requested), not a script. The function should not be interactive. Interactivity can be done in the calling routine, which can read a file or query the user (for the file name or the points).

Test inputs and other stuff. This will require writing a script to call the function (I did not state this explicitly in the original homework – but you had to do something to call your function, even if it was interactive from the command window of Matlab, and the most efficient way to do this is to use a script to call your function as you run it over and over as you debug it). Your test data should exercise the code for the major classes of input and result (such as getting the quadrants correct). **You should turn in all the work and input files needed so that I can unzip your homework and just run your calling routine (include a word file with instructions, or the call, if necessary), which should read your test input data file. This way I can test your routine without having to read the code to figure out the interface, make up an input data set, write a calling script, etc.**

**You should subroutinize (== functionize?) your function.** If you have to do something twice (some stuff, such as checking the inputs, is done 4 times, twice in each function) you should write a subroutine (function) to do it. The subroutines (functions) you need can be in the same file as your main function (in that case they are private to that function – you cannot call them from the command window, only from the function).

Some combined both tasks into a single function. I consider this acceptable, but practically you don’t always need them together, and you can do the combination from the individual ones.

**Do your own work.** It is OK to discuss the homework and ask me or other students for help on how to do things. When I get multiple copies of the same code with the same error (the code is “correct” in that it works – or seems to, it is just doing the wrong thing) it indicates that there is a problem.

**Reuse your work.** If you need delta in the azimuth function, use the function you wrote for delta (don’t recode it in the azimuth routine).

**Comment**, comment, comment, and use whitespace (blank lines) to make it more readable.

Use the following test data set

Earthquake at (lat=0, lon=0), 8 stations 45° distant, spaced 45° apart [(0,45) (30,35.25) {this is approximately 45° distant at 45° azimuth, you can check with Matlab routines} (45,0) etc. to get all 8 positions]. Note that this data set will check the results for all quadrants.

Additional earthquake location (Memphis) and station locations (Chicago, Illinois; New York, New York; Jacksonville, North Carolina; Miami, Florida; New Orleans, Louisiana; Dallas, Texas; Denver, Colorado; Lincoln, Nebraska). Reference where/how you got the locations.