Data Analysis in Geophysics (CERI 7104/8104)
Homework 1 Solutions

My code for the catalog problem, plus the code that I used to test your programs, can be found on the course web page or in my public folder. Each problem was graded based on 8 points for good coding style, and 7 possible points for passing my various tests. The tests were designed to catch most instances of incorrect code, though in a few cases I noticed some incorrect code that the tests did not catch and took off points for that. More details on the tests are given below, and you can look at the test functions for more details as well. If you have any questions on my grading, your code, or my solutions, feel free to come talk to me.

One other thing – no one used a docstring correctly in the code that they submitted. While I did not take off any points if you included the correct information as comments at the beginning of the appropriate function, I *will* expect you to use them correctly when you resubmit this code for the final project. My solution for the catalog problem illustrates how to write a docstring. You are also welcome to speak to me if you do not understand what a good docstring entails.

1. For the catalog problem, I ran seven tests. The first six should lead to expected execution of the code, including correctly handling a catalog that is not sorted, and ensuring that you allowed for negative magnitudes (which are allowed!) while I tested your codes to see if they raised an error if the number of bins was negative.

   One test case looked to see if your code still worked if the minimum magnitude was above the largest magnitude in the catalog. I did not tell you how to explicitly handle this case, so I accepted several possibilities – most of you raised an assertion error (which is an acceptable strategy, in which case I accepted the code not returning a result), while my code allows for this and just returns all zeros. Either would have been fine.

   All of you had working implementations, though not all of you did it in the most efficient way possible. In particular, many of you first calculated the list of bin times, and then looped over every event to check every bin to see which one it is in. Once you found it, you kept track of this, either by incrementing the count for that bin or remembering which bin you found so that you could sum up the results later. This works, but note that you explicitly checked every bin for every earthquake. If the number of earthquakes and bins is very large, this might take a long time as you have to do a large number of unnecessary checks.

   There is a more efficient way that takes advantage of the fact that we know that all the bins are the same size. If all the bins are the same size, then if we know the start time t_s, the end time t_e, and the event time t, then we can calculate $T = (t - t_s)/(t_e - t_s)$. Since t is between the start and end times,

$0 < T < 1$, and it describes the timing of this event relative to all the others in a normalized fashion. Thus, if we multiply T by the number of bins and round down, we will get the bin number to which this event belongs. This way, if we calculate int($N_{bins}T$) we can eliminate the loop over all of the bins and just proceed directly to the correct bin. Note this only works if all the bins are the same length, so we can only use it here because of how the problem was specified.

2. For the travel times, I tested both your `calc_distance` function and `calc_travel_times` for 15 tests in total (12 tests of `calc_distance` and three tests of `calc_travel_times`). The first two tests of `calc_distance` were worth one point each, and the remaining 10 were worth collectively two points. The final 10 tests of `calc_distance` verified that you used assertions correctly to determine if the input was poorly formatted and if the latitude and longitude numbers were in the appropriate range. My additional tests of `calc_travel_times` verified that you correctly handled multiple pairs of sources and receivers, and were worth one point each. You have a chance to correct these errors and any other coding style errors when you resubmit your final project. However, I will not provide solutions for any of the project code.

   I suggest you correct your code right away – you are likely to forget what you did the longer you wait, and when you do your final project you will have to correct 4 pieces of code plus write several new things to tie everything together. Thus, I highly recommend working on this right away. If you lost points on coding style, I am happy to take a look at your code before the end of the term if you want feedback on your revisions. Additionally, you are each allowed to run my test function on your code one time at your request before resubmitting your code at the end of the term. I will let you know how many tests you passed, but I will not provide you with the test code.