**Scientific computing and the student scientist**

As a scientist, few skills like scientific computing have given as great of dividends. In my 20-year publication history, there is not a single one that did not rely on scientific computing to carry out the research. It is hard to stress enough what a necessary skill it is for the student scientist to develop in this digital age. Unfortunately, in my field of the Earth Sciences, student scientists all too often lack – and sometimes consciously avoid developing - this skill, putting them at a disadvantage when they are expected to carry out more rigorous scientific endeavors in the work force or graduate school.

In my experience, when an assignment requires some sort of actual analysis, the go to is Excel. Clearly, Excel can do a number of things well, especially when just a quick and dirty solution is necessary. It is relatively intuitive in that the data are at the forefront, it does quick operations on large lists of numbers, and can plot on the fly. However, spreadsheeting is inherently less flexible than a more algorithmic language. To encourage more familiarity with using computers in scientific work, I have been gradually incorporating more computer based problem sets in homeworks I assign.

I would love to promote learning FORTRAN or C+ as they have distinct advantages of speed, legacy, and being non-proprietary. But, given time and other constraints it works best to reduce expectations to one high level language. I have settled on MatLab as a standard for assignments. As a scripting language, it has a shorter learning curve, it has many intrinsic functions allowing for fewer lines of code to do the same work, and crucially has the capability of high quality visualizations integrated into the functionality. I am also convinced that just getting a student comfortable with scientific computing as a regular tool prepares them for realizing the importance of lower level languages when they later encounter them.

One anecdote that greatly affected my feelings about the differences in languages for scientific computing comes from when I was a post-doc. I was developing a model of seismic attenuation. This required use of a non-negative least-squares algorithm to invert with several hundred parameters and many thousands of data. I found the function *nnls* in MatLab and within 10 lines of code, I could carry out a fairly substantial inversion. However, it took more than three days to complete. For comparison, I found a non-negative least squares function in FORTRAN which, of course, depended on other linear algebra packages and took considerably more programming to develop the same type of results as *nnls* produced. Rather than 10 lines, I had closer to 10 pages of code – but it completed in 20-30 minutes. The exercises I am having my students do are not susceptible to this difference in scripting vs low level language time differences, but it underscores using the right tool for the problem. Even MatLab is not always the right tool, but it is an excellent one for developing the skills of scientific computing we expect of our next generation of scientists.