*Why is computation important for your students and how do you develop students' comfort with computation?*

*Essay by: Michael Cardiff*

As a hydrogeologist, I consider myself one part geologist and one part engineer. Hydrogeologists – like other geologists – must have the practical and qualitative skills to interpret geologic evidence from the natural world. However, much like engineers, hydrogeologists are also required to develop quantitative and defensible models that can make predictions about how a system will behave under changing stresses. For this reason alone, being fluent in computation in this day and age is basically a pre-requisite for being a strong hydrogeologist. More broadly, however, I believe students in our graduate program benefit from an understanding of computation and programming because it encourages clear, succinct, and quantitative thinking. I have heard many faculty in my department profess that “clear writing shows clear thinking.” I believe this is even more true of clear programming, which requires a degree of planning and metacognition comparable to what scientists do as they plan an experiment or design a new tool.

My philosophy in getting students to develop comfort with computation and programming (throughout my MATLAB class) is that learning a programming language is largely similar to learning a foreign language – the structure of the language is natural and logical once you know how to speak it, but one has to develop fluency first through a lot of trial and error. My initial “motivation” presentation in the class focuses on the following mantras, that I believe are equally valuable when learning foreign languages and programming languages:

1. Be goal oriented – that is, start with a problem you need to solve in mind, and then pick up the tools you need to get the job done;
2. Immerse yourself – that is, commit throughout the semester to do your quantitative work efficiently in MATLAB, rather than falling back on crutches such as Excel;
3. Know how to use your resources – that is, think of the MATLAB help files as a “dictionary” to the language, so that you can help yourself and learn through examples;
4. Finally, be experimental and fearless try clicking that button! Try hovering your mouse! Try a function you’ve never used before! If you get an error code, you’ve just learned something. Break things, and then figure out how to fix them.

In following the first mantra, in particular, we expect that students begin thinking about a final project basically on the first day of class, and we attempt to scaffold progress on this final project throughout the semester. Within the first couple weeks of the course, we have students write a “pre-proposal” concept for a problem that they would like to try to solve using computation. Almost every student encounters something in their research that is complicated, tedious, or otherwise difficult to do manually, and we encourage students to think of these as opportunities to enhance their research through programming. Once we have progressed roughly halfway through the course, we then ask students to submit a “full proposal” that describes their plan of attack for attacking their problem, by way of dividing up their larger problem into sub-problems, and presenting a schedule of milestones that they plan to meet along the way. Roughly ¾ of the way through the class, we review students’ progress on meeting the milestones in their proposal, and suggest avenues for changing their project if their initial concept proves to be either trivially simple or exceedingly difficult. Finally, we have students present the results of their final project through both an oral presentation and a written report that includes their software. Throughout, the students are working toward a final goal of a useful, modular, and well-documented tool worthy of (for example) being uploaded to the MATLAB file exchange.

It’s common at the end of the semester that each student has developed a different skillset. Some students have developed analysis algorithms with complex logical structures, error checking and the like. Other students may have spent more time on visualization, and developed useful functions for plotting particular types of results. And other students may have built up familiarity in dealing with importing or exporting specially-formatted text files that are pre-requisites for later analysis. Even though every student may come out with different skills, it’s our hope that students have developed comfort with the particular tools that are most relevant to their own interests and research.